2012.08

PROGRAMMER 程序员

我们的开源

17 艾伦·图灵
如谜的解谜者

20 开放中创新
体验 Google I/O 2012

80 如何做好技术布道

86 《酒店达人》设计谈

90 多角度着眼未来世界Online

95 从零开始学游戏编程
可视化编程游戏开发工具学习指南

100 结合场景的HBase 性能分析

105 CloudStack 架构详解

110 《数学之美》作者吴军
把握本质规律

113 SNS 中的文本数据挖掘

116 Facebook Folly
源代码分析

127 Mac OS X 文件系统的来龙去脉

134 “开放源代码运动”领袖
Eric Raymond

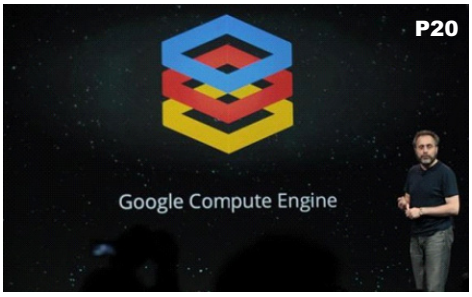
ISSN 1672-3252

9 771672 325128 08>

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents



资讯

5 外刊速递
8 网文精选
10 新闻
12 新产品新工具
14 程序天下事
17 艾伦·图灵：如谜的解谜者
20 开放中创新
——体验Google I/O 2012

封面报道：我们的开源

本期封面报道聚焦中国本土开发人员和团队主导和参与的开源项目，从开源项目本身、企业与个人的参与情况、开源社区建设及开源法律知识等多方面，全景式地解读中国的开源现状。

29 大势所趋话开源
——中国开源现状分析
33 乐趣与高效
——淘宝章文嵩谈LVS与商业公司参与开源
36 OpenResty发展之路
——OpenResty项目创建者章亦春专访
39 开源是种生活方式
42 从BlenderCN，看开源社区建设
45 你必须了解的开源法律知识
48 OceanBase：淘宝开源海量数据库
52 NiuTrans：开源统计机器翻译系统
56 Jscex：回归JavaScript的异步流程控制类库
60 Coreseek：中文检索系统
64 Muduo：多核时代的C++网络编程

管理

70 人才招聘新趋势：垂直性的社交网络
——pongo网（庞果网）CEO李炯明专访

72 分布式敏捷团队的经验分享

76 产品经理让创业基因更炽烈
自立门户、创立一家属于自己的公司只是创业的一种表现形式，真正的创业应该是一种精神，一种生存状态，是开创一种新的格局。本文将重点讲述缘何炽烈的创业基因可以让产品经理的工作更加卓越。

80 如何做好技术布道

移动

85 接地气

86 《酒店达人》设计谈

89 开放平台，有所为有所不为
——UC首席运营官朱顺炎专访

90 多角度着眼未来
——《世界Online》策划总监罗维专访

93 游戏模式与玩家需求契合度的分析

95 从零开始学游戏编程
——可视化编程游戏开发工具学习指南
本文作者是一位游戏设计师，文中他将结合自身实践分析“门外汉”学习编程的难点，并分享利用可视化编程游戏开发工具学习游戏编程的经验。

云计算

100 结合场景的HBase性能分析

105 CloudStack架构详解
CloudStack是提供云计算服务的完整解决方案。它整合多种硬件资源，为用户提供透明的云服务，负责将虚拟资源映射到具体的硬件资源，并在用户间提供有效的安全隔离。

2012年08月刊　总第238期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹄翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

技术

110 把握本质规律
——《数学之美》作者吴军专访

113 SNS中的文本数据挖掘
在2012年6月刊中，作者给出了中文字符串能够成词的若干标准，从而实现了无知识库的自动抽词程序。在这篇文章中，作者将在抽词程序的帮助下，以词语为单位，在各个维度上挖掘人们在社交网络中的用词动向。

116 Facebook Folly源代码分析（下）

122 谁动了我的句柄
下班到家，有一份快递摆在案头，一看地址，来自千里之外。打开层层包装，是一张光盘，装光盘的纸袋上七扭八歪地写了两行字：“程序崩溃，偶尔出现；困扰已久，务请帮忙”。明白了，是朋友要我帮忙分析的转储文件到了。

百味

127 Mac OS X文件系统的来龙去脉（下）

130 新书上架

132 GEEK

134 “开放源代码运动”领袖：Eric Raymond

企业专栏

24 中国互联网企业的研发之路
——与腾讯研究院院长郑全战一席谈

26 做有中国特色的云计算
——IBM大中华区系统与科技部电信行业及OEM业务总经理侯淼专访

68 应对大数据，数据库在行动
——来自对IBM信息管理团队的采访

78 技术引领体验
——网易邮箱技术总监向东专访

Wired

2012.7

提速百分之一秒：打造更优秀的奥林匹克健将

由红牛赞助的22位科学家和技术人员正趁着运动员热身对设备进行调校。Optojump监测系统通过40个沿着跑道分布的动作捕捉摄像头，测量每一次跨栏后脚部触地位置和精确触地的时间，再加上安装在跑道旁对运动员进行变焦成像并以1600帧/秒速度进行拍摄的高速Phantom Flex摄像机——这就是美国跨栏新秀Lolo Jones的奥运备战阵容。Jones和教练、科学家们一起观看视频，了解自己每次跨栏后的前腿落地速度。“我们发现我的前腿落地速度尚未达到极限……我正在尝试加快一点每次跨栏后的落地速度，也许每次落地再远1英寸。经过10次跨栏，就是10英寸，倘若百分之几秒就足以分出高下，那么10英寸是很长的一段距离。”

对于顶级运动员来说，传统的训练方式已不够给力。从很好到最好，需要最优化每一分表现，百分之一秒的优化结果也值得争取。因此，除了依靠教练和队友，运动员还需要与生物机械学家、生物学家、心理学家、营养学家、体能教练、康复专家以及统计分析师进行合作。简言之，科学已成为运动员攀登新高峰必不可少的装备。过去，教练靠直觉和训练大幅度提高运动员的成绩，现在，科学家接过最后一棒，帮助运动员臻于完美。21世纪面临的挑战不同以往，经过20世纪的飞速发展，那些可以轻易实现的体育进步早已实现。胜利者的领先程度越来越小，人们日益发现，有助于运动员的工具不是在举重室，而是在实验室。许多运动变得像赛车一样——为了胜利，驾驶技能和科学技术缺一不可。

体育科学家Giuseppe Lippi在一份历年世界纪录分析报告中指出，未来“运动成绩将越来越不受运动员天生生理机能的约束，而更多要受科学和技术进步的影响”。别再惦记着招募最好的运动员了，倘若诚心想打造一支了不起的运动团队，现在就该动手招募最好的博士了——澳大利亚运动学会（AIS）已这么做了，而且成绩斐然。

私人无人驾驶飞机之兴起

2011年，巴黎航空展最热门飞行器的行列中出行了无人驾驶自动飞机的身影，它们被当做未来武器出售，售价从几万美元到几百万美元不等。2012年5月，在北京举办的中国国际模型博览会上，中国制造商展出了各种最新颖、最炫酷的玩具，深圳DJI创意公司出售的无人驾驶飞机展示了军用飞机一般的性能，售价却低于1000美元。然而，这些中国公司还是碰到了售价更便宜的竞争对手——世界各地的业余爱好者在网络社区免费分享的各自的设计。

如果你在美国仰望天空，很可能就会看见一架这样的无人驾驶飞机从头顶掠过，它们体积小、全自动、超便宜。能飞行的智能手机——这是许多人对这些无人飞机特点的概括。除了轰炸，它们具有军用飞机的一切功能。尽管在技术上不主张这些无人驾驶飞机投入商用，且要求飞行高度在400英尺以下、目光可及、远离人口密集地区和飞机场，但美国联邦航空局正准备正式批准无人驾驶飞机从2015年开始投入商用。

业余爱好者用他们的无人驾驶飞机做些什么呢？就像早期的个人电脑一样，目前的主要用途还是实验，纯属娱乐。不过，随着无人驾驶飞机越来越复杂、越来越可靠，其实用价值开始显山露水。电影业已处处可见将遥控直升飞机作为摄像平台，它们比摄像机升降臂拍摄范围更大，比人工驾驶直升飞机更便宜、更安全；农夫用无人驾驶直升飞机绘制飞行图，优化浇水、施肥，进行收割管理；其他科学用途也数不胜数。2007年，无人驾驶飞机爱好者成立了网上社区DIY Drones，如今成员已接近3万。每个月约有1000架新的私人无人驾驶飞机升空，这个数字可以与世界各顶级航空公司的无人驾驶飞机销量（但不是销售额）相媲美，而且私人无人驾驶飞机行业的发展速度更快。

因特网源起于军用，民间却通过克隆创建了用于实现自己目的的网络；无人驾驶飞机业余爱好者社区也希望出现同样的盛况，让无人驾驶飞机非军事化、民主化，充分发挥无人驾驶飞机的潜能。

倾情未来——X大奖创立人 Peter Diamandis

在Peter Diamandis广阔的视野中，小小地

球远非世界的全部。他最初成立的一批公司（共12家），以开拓人类殖民太空道路为目标，充当着社会与科技进步的加速器；此后，作为X大奖基金会创办人以及奇点大学（Singularity University）合作创办人，全力以赴迎战地球上最紧迫的难题。

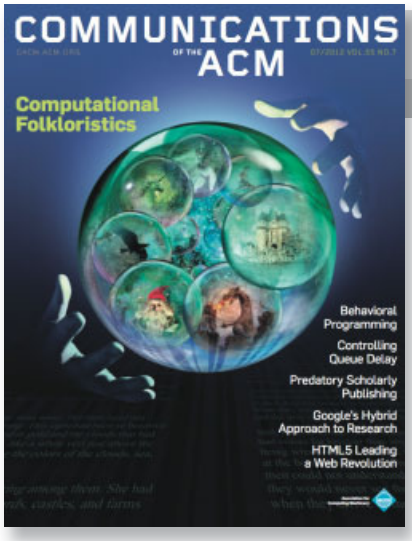
Diamandis伴随阿波罗登月计划度过儿童时代，从小到大他都在盼望美国政府殖民外太空。可是，几十年来NASA始终羞羞答答，这终于使他确信：飞离地球的唯一办法是建立私人航天工业。他大胆设想重新拾起20世纪早期的理念：设立丰厚奖金，迎来航天里程碑。民用航空将通过一次次竞赛、一次次创新成长起来。

这个想法是对的。安萨里X大奖（Ansari X Prize）高达1000万美元的奖金迎来了首次私人天际往返飞行，揭开了一系列航天竞赛的帷幕。此后，Diamandis如法炮制，将这一措施用于燃油效率、溢油修复、保健成本等问题的解决，结果获得了一系列突破性成果，引起了一阵创新激励竞赛的小繁荣。

他很快意识到，那些促使一小组业余爱好者制造出太阳能登陆舱的力量同样能鼓舞聪明的理想主义精英们解决与地球有关的问题。为此，他创办了奇点大学，这是一所在暑假开课的大学，每学期历时10周，如今已经进入第四个年头，其目标是培训下一代领导人利用飞速发展的技术解决Diamandis所谓的“人类重大挑战”。

再看看他最近的行动：四月，名为“行星资源”（Planetary Resources）的新公司开张，它将在从地球旁边徐徐飞过的小行星上开采铂金、水和其他可提取物。Google公司拉里·佩奇、埃里克·施密特，还有罗斯·佩洛特，以及前微软首席软件主管西蒙尼·卡罗利均在这家新创立的公司投入大量资金。

Diamandis在与Steven Kotler合著的《富足：未来比想象更美好》（Abundance: The Future Is Better Than You Think）中提到：技术托起狂野的梦想，把我们送入星河，把我们送入空前繁荣的时代。事情正是如此。



Communications of the ACM
2012.7

Google的混合式研究方法

通过关注创新、服务模型、大型用户社区、人才队伍，以及计算机科学研究的发展属性，使Google形成了一种“混合模型”。在这个模型下，Google模糊了研究和工程设计之间的界线，鼓励团队寻求恰当的平衡，同时了解这种平衡时刻在变。Google还随着需求的变更，在人才和项目转移上保持相当好的流动性。因此，即使在研究工作所占比例远高于工程设计的领域，Google所建立的“研究队伍”也不像其他组织中的研究队伍一样，往往与工程设计工作脱节。例如，研究队伍也会运作大型生产系统。总体上说，当Google感到相对高风险的项目有可能促成深远的技术变革时，就开展研究工作。此外，通过Google的产品和服务，以及通过学术型研究社区，研究工作还有可能影响全世界。Google意识到，通过积累有价值的反馈、教育未来新员工、提供合作，以及催生额外工作促进重要成果广泛传播，往往于Google有益。

Google研究模式实例

1. 以产品为中心的团队完成高技术项目，借由项目获得的成果改变技术现状。这是Google最主要的模式，它模糊了研究和开发之间的界线。

2. 由研究团队完成项目，最终形成新产品或新服务。在这种模式下，会在进行研究之后展开基于该研究的生产服务。Google Translate和Voice Search都是这种模式的具体例子。

3. 由研究团队完成的项目，最终形成新理念和新技术，这些理念和技术随后应用于现有产品或服务。这是一种传统的研究和开发模型。

4. 由工程设计团队和研究团队合作完成项目，随后由该工程设计团队使用。这种模式是研究和开发团队的合作整合模式。

5. 由工程团队完成研究项目，然后移交给研究团队，最终成为上述2、3、4类项目。当一个项目不仅对于一个特定工程

设计团队意义重大，而且在更大范围内具有重大意义时，这种将项目从工程设计团队移交到研究团队的模式就成为一种为项目提供更多时间、更多资源的重要机制。

掠夺性学术出版

学术出版是一类十分独特的商业活动。在普通商业活动中有两个组成单位：卖方和买方。在学术出版中也有卖方和买方，即出版社和研究型图书馆。但学术出版还另有两个单位存在：急于免费发表作品的作者和起到把关作用的编辑或校对。他们的参与有着多种多样的原因，有时是为了得到经济收益，但大多数时候是为了尽责任、赢声望。

作为一种商业活动，出版社为了让学术性出版获得成功，必须说服图书馆订购它们的出版物。过去几年，图书馆的预算日益收紧，十分不情愿增加订购量，实际上的趋势是省钱、省钱、再省钱。若要说服图书馆将出版物添加到订购书目中，必须以质量取胜。图书馆的这种不情愿成了维持学术出版质量的重要力量。

但近来的趋势打破了出版社、图书馆、作者、编辑之间的这种平衡，出版社不用再说服图书馆订购书籍。原因是什么呢？一方面，数字出版走向合法，许多高质量的出版物只出版数字版本；另一方面，自助出版的普及使得作者付费模式得以盛行，在这种模式下，出版社的收入来自于作者费用，而不是图书馆订书费。虽说作者付费模式从根本上说没有什么错，图书馆的退场却造成一种不平衡的系统。出版社和作者都希望出版，而图书馆却无所谓；编辑和校对试图稳住质量，可是面对急不可待的出版社和作者，他们的力量黯然失色。学术出版新经历的这种不平衡现象助长了“掠夺性”出版，掠夺性出版的目的不是促进学术，而是创造利润。在某个非正式名录中，掠夺性出版社和期刊条目已多达50条（见http://scholarlyoa.com/）。

《ACM通讯》主编Moshe Y. Vardi指出，并非所有商业出版社都是掠夺性出版社，但出版社首先都是逐利的，出版社和作者之间的利益冲突由此而生。学术出版的未来在于联合出版，出版社、作者、编辑和校对作为这种出版方式的参与者共同为学术作贡献。

Figure 1. A schematic view of the execution cycle of behavior threads using an enhanced publish/subscribe protocol.

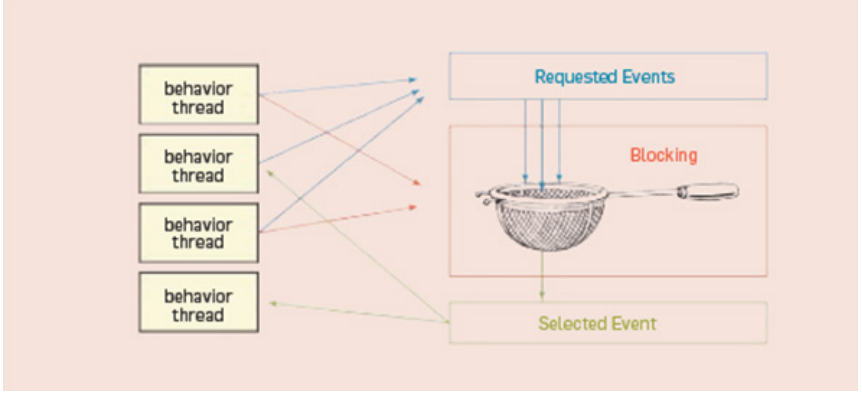


图1　使用增强的Publish／Subscribe协议的行为线程执行过程图

行为编程

“行为应用程序”这个术语指的是由独立组件（称为b线程）构成的软件，经由增强的Publish/Subscribe协议生成图1所示的事件流，每一个b线程都是一个过程，与其他线程并行运行。当一个b线程到达一个要求同步的点以后，会等待所有其他b线程到达各自流程中的同步点。在每个同步点上，每一个b线程都会指定三个事件集：请求事件、等待事件、锁定事件。当所有b线程都位于同步点上时，就会选择一个事件，该事件至少由一个b线程进行请求，且未被任何b线程锁定。

行为编程原理可在各种语言和编程方法中轻松得以实现，成为这些语言和方法的组成部分，除了带BPJ包的Java，研究者还在功能性语言Erlang和Shimony中实现了这种机制，并在使用C语言的PicOS环境中进行实践。在可视环境中的应用除最初的Play-Engine外，还包括PlayGo、SBT等。但“请求事件”、“等待事件”、“锁定事件”等习语可能会发生变化。

行为编程提供了一些实用的编程机制，可以对某个行为，而不是对所有其他相关行为实现隐性的、间接的控制，不会将来自控制或限制模块的显性内容引用到受控基本模块中。此外，在行为编程过程中，所有系统行为都得到平等对待，基本行为和高级行为之间不会存在差别。研究者相信，行为编程可用于实现对称的具体情况，作为对当前盛行的、将基本代码与具体情况区分对待的非对称方法的补充。

（感谢译者李芳支持）

软件库存

文 / Joel Spolsky　链接：http://bit.ly/NLlqN8

本文是Joel大侠的新作（对，就是《Joel说软件》的那个Joel），以“库存”为类比讨论如何改进软件开发的效益。

想象一下面包作坊：机器中间摆放着一桶桶芝麻、一坨坨面团、一袋袋面粉，还有一箱箱做好但尚未运走的面包。

这些都是库存，库存可不是免费的。假如平均每天储备150吨面粉，那要花费多少钞票？库存还有其他成本，面粉可以多放些时日，面包一过24小时就卖不掉了。那为什么需要库存呢？因为“耗尽”也会带来成本。假如订购一批芝麻需要两天时间，一旦芝麻用光，便有两天时间不能做生意了。库存提供了缓冲，防止生产过程的任一环节中断。

可见，库存没有不行，但也不是越多越好。一些现代算法可以计算这一缓冲的最优值。

为何关心起这个来？因为软件生产过程中也有几处主要的“库存”堆积点——在这些节点，事情会堆积起来，浪费大量的时间和金钱。

把你的“想法”想象成原材料。一种想法要经历多个流水环节，如决策、设计、实现、测试、调试、部署等，最终才能作为就绪的特性交付客户。在环节之间，库存会堆积起来。比如，程序员每完成一部分代码便交给测试员检查。在任何时间，总会有一定数量的代码在等待测试，这便是库存。

“库存”代价高昂。有可能长达6~12个月的工作堵在流水线上而没有到达客户手中。而这会导致严重后果，造成的差别就像头前领跑的iPhone和不断追赶的Windows Phone那样。

我们来看看软件开发中最容易堆积库存的三个地方。

特性候选单：产品设计时总会有新想法出现，而实现这些想法的速度永远赶不上把它们想出来的速度。你把它们写下来，它们就成了候选特性。问题在于有90%的候选特性你永远都不会去实现，于是每次你把这些永不实现的特性记录下来，考虑一番，设计一番，甚至集体讨论一番，都是在浪费时间。

Bug数据库：这本来是个好东西，然而在许多公司里，不遗漏任何Bug报告的目标反而导致了“Bug破产”：某天一觉醒来，你发现库中还有3000多个Bug处于开放状态，一些古老的已没有实际意义，还有一些已无法重现，更多的则太小，不值得修正。可是，已有成年累月的工作消磨在这些Bug报告的准备和维护中了。

未部署的特性：这是库存中代价最高的一种。这些特性本来可以为你赚钱，却一直放在货架上没卖出去。马路对面男孩早已在使用另一款产品，里面有完全相同的特性。

软件开发防错指南

文 / Aaron Swartz　链接：http://bit.ly/MNI5E1

本文作者借用“精益生产过程”中的防错（pokayoke）思想，从需要、想法、员工聘用、假设前提、团队、开发、软件架构、部署和发布等多个环节全方位探讨了软件开发中的防错方法。

需要

好的项目都从一个需要开始，需要必须敏锐，要针对用户渴望解决的问题，你自己也要感受到这种需要。最理想的情况是：它是你自己的需要，源于你的亲身经历；退而求其次，它是你亲身去体验一种生活方式从而激发的需要；再其次，至少你能静心观察一个有这种需要的人，设身处地地想象他的感受。

想法

只有需要还不够，还要有想法来实现这一需要。要客观地考察自己的想法，它真能实现这种需要吗？许多想法之所以不好，就是因为它们不能真正实现需要。先有需要而后有想法，要记住这个次序。不能反过来，先有想法，然后因为太喜欢这种想法，想各种貌似正当的理由臆造需要。当然，一个想法可以解决多种需要。

说说iPhone。有人会觉得乔布斯不过是有过一个泛泛的、看起来不错的念头，然后碰巧这个念头解决了多种需要。事实并非如此。iPhone项目启动时，乔布斯就坚称它能满足三种需要：宽屏的视频iPod，活力四射的网络沟通设备，好用的手机。

员工聘用

招聘一名员工时要考虑三个问题：此人

是否聪明，能否做出东西，是否易于相处。聪明但不能做事的人可以做朋友但不要做员工，只要能跟他们交流聪明想法就够了。能做事但不聪明的人效率太低，常常用笨办法解决问题，聪明的人会看不下去，结果放下自己的工作去帮忙。无法相处的人就是无法相处，大家都会说“反正只是工作，又不要做朋友”，

发布

不要搞好莱坞式的产品发布会，软件跟电影不同，软件不在院线发布，你也不用担心一周后会没人放映。你可以长年累月地推进一个软件项目，持续不断地壮大客户群。

聪明人之我见

文 / Tommy MacWilliam　链接：http://bit.ly/NXReJh

来到哈佛，便有了极佳的机会接触许多聪明人。我说的“聪明人”不是指那种知识竞赛中的胜出者，而是各种领域的杰出学者。

首先说明我自己并不是这样的人。在那些聪明人身上，我总能找出自己无论多么努力都无法企及的方面。这没什么，我可以不必那么聪明。但如果不能从这些人身上学到点什么，那就太对不起这些机会了。

从这些人身上，我还真的注意到了非常重要的一点：他们都喜欢提问。

如果有人向我解释一件事情，我只会点头，显得我了解对方在说什么。如果不了解，回头我会去Google一下。总之不想被人当成傻子。

聪明人不一样。不论懂还是不懂，他们都问。我清楚地记得有那么一次客座讲座，解释完一个概念，讲师问大家有没有问题。我觉得那是个直白的概念，其他同学也都点头，表示没有问题。然而还是有人提问了，问题来自一位显然了解过相关议题的终身教授。当时我没觉得怎样，或许还觉得自己比那个教授聪明，因为我明白的一个概念他不明白。现在我确信教授问那个问题绝不是像人们想的那样，仅仅为了让讲师感觉良好，启发一些讨论，或者其他什么。提问的语调和聆听的专注都显然说明教授很虚心，答案很重要。

聪明的人不仅在不懂的时候提问，在整个世界都认为他们懂的时候，他们还是会提问。他们挑战人类认知的极限，认为一切皆可能，不找到能让自己坦然接受、完全理解的解释就决不罢休。

聪明的人挑战一切。（这也是一个聪明人告诉我的。）

也许有一天人们也会把我当作聪明人，但现在，我还要不断向他们提问。

实现简单的贝叶斯分类器

文 / Alexandru Nedelcu　链接：http://bit.ly/MLBQ3q

先举几个分类器（classifier）的用例：垃圾邮件检测、第一语言自动判定（如Google翻译）、表情分析。

通常专用技术做得更好，然而本文介绍的简单贝叶斯分类器是通用的，它易于实现，且对多数应用来说都已足够好。此外，虽然专用算法能带来更高的准确度，但在一般情况下，更好的数据加上可以自行调整的算法，能以更低的代价获得更好的结果。

条件概率和贝叶斯理论

这里有一个示例。假设我们有如下统计信息：总共74封邮件中，30封是垃圾邮件（S）；这74封邮件中有51封含有字母P开头的某个单词（P）；含有该单词的邮件（P）中有20封已被标为垃圾邮件（S）。

问题：如果刚刚收到的一封邮件也含有这个单词，它是垃圾邮件的概率为多少？

事件S、P显然是独立事件，于是，你可以根据贝叶斯理论的简单形式来求解：

```
P(S|P) = P(P|S) * P(S) / P(P)
= (20/30) * (30/74) / (51/74)
= 0.39
```

贝叶斯方法

上面是个简单例子，结果不难理解。但如果增加更多过滤单词，公式马上会变得复杂起来，然而，如果我们简单地假设这些单词的出现彼此完全独立，公式还是可以相对简单。

要分辨出一封垃圾邮件，需要根据邮件中出现的单词来计算条件概率，简单贝叶斯方法就是我上面描述的这种方法：假设单词的出现彼此完全无关，从而简化处理过程，降低复杂度。这种方法的确有它的缺点，因为某些单词之间显然有联系。但那只说明结果的准确性欠佳而已。

（感谢译者王江平支持）

人才收购

“请大公司冷静考虑收购，若非要如此，请像Facebook对待Instagram那样，让他们继续专注于自己的产品。”——2012年7月20日，流行邮件客户端应用程序Sparrow被Google收购，Hacker News上有人呼吁道。

从好的方面来看，可能意味着除了iOS和OS X用户外，Linux、Android和Windows用户未来也可能免费体验到这款创新产品。但这起收购更多的是引起了老用户的担忧，他们担心发生在Tweetie身上的故事将会重演，原本出色的应用将从此变得平庸。

同一天，开发了Pulp和Wallet的Acrylic公司宣布被Facebook收购。Instapaper的作者Marco Arment将这几起收购称为“人才收购”：这些公司被收购，并不是母公司为了继续发展其原有的产品，而是为了将这些开发者收入麾下，让他们开发母公司的产品。通常情况下，被收购的产品将会在不久后被关闭，或者被遗弃。

Instapaper也曾收到过大公司抛来的橄榄枝，但谈判从未深入下去。一方面是因为Instapaper的成长还很健康，Marco Arment不希望它被关闭；另一方面，大公司开出的报酬也没有足够吸引力——谁愿意为即将关停的服务花大笔钱埋单？

如果你还希望从这些创业公司的应用中品尝到惊喜，那么请尽你所能善待它们，如果开发者能从自己的商业中获得足够的成功，大公司的邀请就不再那么有吸引力了。

抄袭

2012年7月13日，iOS平台ARP游戏《Torchlight》首席设计师Travis Baldree在Twitter上指出，一款名为《Armed Heroes Onlice》的游戏直接使用了《Torchlight》的设计元素。他将两款游戏的人物与场景设计进行了对比，结果显示，两款游戏中各种人物和场景惊人相似。虽然最后《Armed Heroes Onlice》遭到了App Store的处理，但让人疑惑的是，在对应用的审核中，苹果到底做了什么？

短短三日后，移动游戏开发商MagicBone在新浪微博上宣布，他们开发的《Pyramid Run》在北美免费榜冲至第一名。相信很多人看名字就能猜到这是一款对《Temple Run》的抄袭之作。《Pyramid Run》的运行并不流畅，而且充斥着影响体验的广告，得到了大量一星的评价。这款应用能登上北美免费榜第一名的成绩可以称得上是“奇迹”。究其原因，可能是它的名字误导了部分用户或有人抱着猎奇的心态围观。

长久以来，《Temple Run》的抄袭之作如潮水般在App Store上涌现，这款连名字都很像的App却能在苹果的眼皮底下达到如此成就，其中的缘由也令人费解。

值得注意的是，上述两款应用背后的开发商EGLS Technology和MagicBone都来自中国。中国开发者的创新能力历来遭人诟病，这两款应用的出现又增加了一条例证。虽说是抄袭之作，但想必也耗费了不少精力，最后的结果却难令人满意——既浪费了资源又坏了名声。开发者们是否该问问自己，这样做真的值得吗？

“写代码会让你感到卓有成效，编写文档会让用户感到卓有成效。善待你的用户，请用心编写文档。”

——Go语言的主要设计者Rob Pike在Twitter上谈到了文档对于用户的重要性。优秀的软件不单功能出色，也该有良好的文档。

“索尼和微软还在就运算性能方面的问题争论不休，但最终，这些都将不再重要。因为一旦HMD技术成熟，体验将会完全不同。”

——id Software创始人John Carmack认为，即使游戏主机的性能再强悍，对于玩家的体验，也不过是锦上添花，而头戴显示器将会给游戏领域带来变革。他的实验性设备已经获得了初步成功。

“我们经常使用的一种技术就是人工模拟站点的负荷。”

——Rajiv Eranki曾负责Dropbox的扩展性工作，经历了用户数从4千到4千万的激增。他正在撰写系列文章，讲述Dropbox的可扩展性设计经验。

“Musk在另一个维度工作，这些工作的真正意义，甚至不是我们这一代人可以评价的。”

——在互联网、航天和电动汽车领域分别取得阶段性成功后，已经有许多人认为Elon Musk的成就将超过乔布斯。

“关键并非Objective-C本身，而是苹果产品及系统很成功，让编程语言因此受惠，若苹果平台当初采用另一套语言，也会出现相同的效果。”

——在最新一期的TIOBE排行榜中，Objective-C超过C++，登上了热门编程语言第三名的位置。在谈到这些变化时，移动应用程序开发平台Appcelerator CTO Nolan Wright如是说。

“2011年7月至12月，美国政府共索取用户资料6321次，内容包括电子邮件通信、文件、浏览记录，甚至是设立账户时的IP地址。”

——Google日前公布了《企业透明报告》，这些最新数据展示了互联网公司将用户隐私资料交给美国政府的频率。

“因此，有必要存在一种系统，既能模拟传统乐器，又可以提供友好的界面，让使用者输入音乐，并看到结果。”

——美国专利商标局公布了一组苹果新获得通过的专利，内容包括根据按键敲击节奏修正输入内容、双影像感应器配件和GarageBand输入界面。

CSDN十大资讯

2012年6月20日–7月20日

01 微软正式发布Windows Phone 8

作者 / 张勇

新系统带来了诸多升级，但老用户无法升级到最新的Windows Phone 8。这意味着接下来的几个月，用户购买的任何Windows Phone手机都无法升级到下一个版本，这对许多支持Windows Phone的老用户而言是当头一棒，也给本就处于艰难困境中的诺基亚带来了更大的危机。

02 闰秒让部分互联网企业很受伤

作者 / 张勇

国际地球自转服务组织和国际地球时间局宣布，在2012年6月30日23:59:59增加一秒，记为23:59:60，这增加的一秒，让一些互联网企业很受伤，它们的服务因此而受到影响。受影响的网站包括Reddit、Mozilla、FourSquare、Yelp、LinkedIn和Gawker等。

03 Linux Kernel 3.5发布

作者 / 王然

经过7个RC候选版，Linus Torvalds宣布Linux Kernel 3.5发布，网络、电源管理和安全性能都得到了提升。新增特性主要包括：支持EXT4文件系统元数据校验，用户空间探测器Uprobes，Android风格的自动挂起，基于Seccomp的系统调用过滤，TCP连接的检查与修复等。

04 雅虎前员工给新CEO的10条建议

作者 / 刘江

Google的第一位女工程师Marissa Mayer出任雅虎的新任CEO。刚从雅虎辞职不久的Sriram Krishnan给出了10条建议，包括裁掉1万人，全力聘请优秀产品经理和工程师，组建精锐部队，提出人们真正能理解的新愿景，全面改进公司内部IT水平，想办法让媒体闭嘴等。

05 新型Android病毒感染10万智能机

作者 / Jon Russell

一种名为MMarketPay.A的新型Android扣费病毒在中国已感染了10万多部智能机，波及9家应用商店。如果用户从应用商店下载了受感染的应用，它便可以绕过中国移动的SMS安全措施进行登录、下载内容并下单，而用户毫不知情。

06 官僚主义拖累科技巨头

作者 / Kurt Eichenwald

本文由乔治·波尔克奖得主 Kurt Eichenwald 撰写，他依靠数十个采访和公司高管的内部邮件，分析了美国历史上伟大公司之一——微软。为大家提供了鲍尔默执政十年下的微软前所未有的视图，同时也诠释了苹果为何只靠一部手机就击败了微软所有产品的收入。

07 HTML5打造：Firefox OS界面曝光

作者 / 魏兵

不久前，Mozilla基金会的“Boot to Gecko”改名为Firefox OS，正式加入移动操作系统的战场。Firefox OS中，没有所谓的“原生应用”。无论打电话、发短信、玩游戏，使用的都是HTML5技术。根据调查，已经有不少开发者表示愿意为Firefox OS开发应用。

08 Rovio新物理游戏“阿力”

作者 / 杨依帆

Rovio推出了新物理解谜游戏《神奇的阿力》，这也是继《愤怒的小鸟》之后Rovio发布的首个“非小鸟”的游戏。Rovio希望向大家证明，他们不是只依靠“小鸟系列”存活的公司。在发布《愤怒的小鸟》之前，Rovio已默默无闻地制作了51个游戏。

09 App Store付费系统被破

作者 / 张宁

俄罗斯黑客Alexey Borodin盯上了苹果App Store的应用内付费系统。利用苹果的交易漏洞，攻击者可以免费下载所有的内付费商品。此举可能严重破坏App Store中的货币制度。受到威胁的不仅仅是苹果公司，开发者也将蒙受经济损失。

10 首次出现季度亏损，微软衰退了吗？

作者 / 张勇

7月20日，微软发布了第四季度财报。报告显示，微软出现了创建历史以来的首次季度亏损——净亏损4.92亿美元。但实际上，微软整体收入仍在增长——微软并没有衰退。亏损的原因主要是2007年收购网络广告公司aQuantive失败，因此减记62亿美元。

Kickstarter 24小时募集资金最多的10个项目

Kickstarter是世界上最大的创意项目资金提供平台，它致力于支持和激励创新、创造和创意性的活动。其意义在于将资金链绕过大公司和政府，直接将创业者与用户连接在一起。

01 OUYA

Android开放平台电视游戏机OUYA仅8个半小时就筹资百万美元，将Kickstarter的诸多纪录抛在身后。OUYA主机将在2013年3月正式发布。

02 《Double Fine Adventure》

因创造《猴岛小英雄》著称的Ron Gilbert和Tim Schafer试图打造一个古典风格的冒险游戏，他们将创意放到了Kickstarter，原计划筹集40万美元，实际收到了300万美元。

03 Pebble

尽管智能手表几年前就已经面世，但似乎直到Pebble出现，才真正抓住了用户的需求。它拥有一块分辨率为144x168的E Ink显示屏，电池续航可以达到7天。

04 《Wasteland 2》

《Wasteland》首部作品于1987年登陆Mac，次年移植到PC，它影响了包括《辐射》在内的多部作品。《Wasteland 2》制作团队包含很多原班人马，预计2013年10月发布。

05 《Shadowrun Returns》

游戏开发商Harebrained Schemes将把Shadowrun Returns搬上平板和PC，并会开放其关卡编辑器，使粉丝们能够在Shadowrun世界中创造自己的舞台。

06 Amanda Palmer

歌手Amanda Palmer为自己的新专辑募集了100万美元。音乐家和艺术家们创作都离不开资金保障，Kickstarter为他们提供了新的收入来源。

07 《The Icarus Deception》

营销大师Seth Godin将他尚未动笔的《The Icarus Deception》也放在了Kickstarter上，目标是募集4万美元。最后的结果是28万美元。

08 Elevation Dock

以往的iPhone底座大多因为质量过轻，会随着手机被一同提起来。Casey Hopkins设计的Elevation Dock使用整块铝合金车削，既美观又实用。

09 Penny Arcade

Penny Arcade是一个老牌在线漫画网站，两位创始人希望通过募集资金，代替原来广告商的展位，让所有网站用户能看到相对干净的页面。

10 gTar

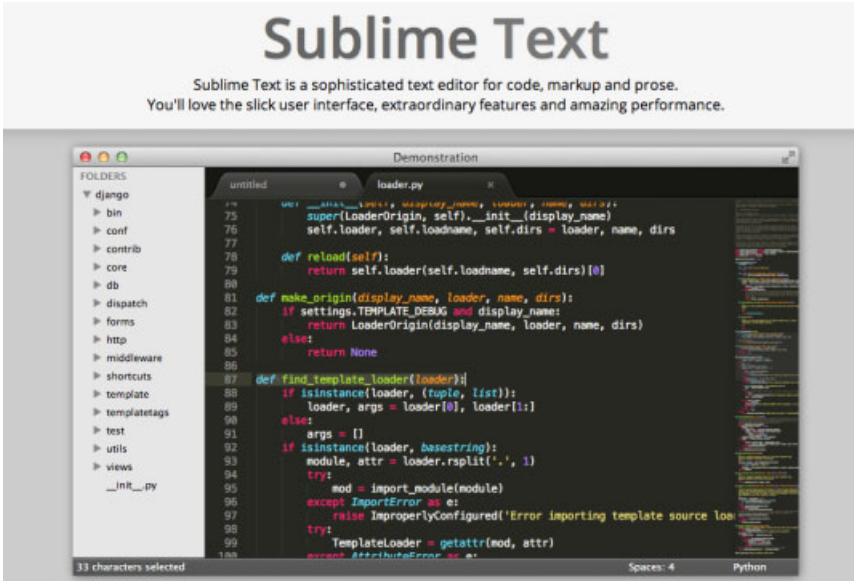
学吉他不是件容易事，gTar就是为了解决这个问题。gTar面板的一部分被挖空，用于插入iPhone。通过gTar应用，你就可以对整个吉他进行控制。

跨平台编辑器Sublime Text 2.0发布

澳大利亚程序员Jon Skinner从2005年底利用业余时间开发Sublime Text。2007年后，他辞去了Google的工作，专心进行这款编辑器的开发。

早先版本的Sublime Text并没有引起很多关注，但在2011年Sublime Text 2 Alpha版发布后，逐渐开始吸引越来越多的程序员关注。根据Hacker News的统计，Sublime Text已经是使用第二多的编辑器（第一是Vim）。Sublime Text的核心使用C++编写，并将Python作为扩展语言。之所以选择Python，主要是因为它简单易学、有丰富的库和更大的用户数量。

Sublime Text 2.0的主要新特性包括：OS X平台上支持新的Retina Macbook Pro，正式支持ODB，新增了Quick Skip Next机制，增加了文本拖放功能，以及更完善的代码补全功能等。



GNU GRUB 2.00发布

GRUB是GNU项目的启动引导程序，允许用户在一台计算机内同时拥有多个操作系统，并在计算机启动时选择希望运行的操作系统。GRUB还可用于选择启动同一操作系统的不同版本内核，也可用于向这些内核传递启动参数。经过十年多的开发，GRUB 2.00终于发布。

GRUB 2.00的主要新特性包括：加入官方主题；支持通过显示器EDID信息读取视频模式设定；新的EHCI、AHCI、ESCC、EFI驱动；新的文件系统和磁盘类型支持，包括ExFAT、LZOP、Squash4和RomFS等；支持引导FreeDOS和Plan9；支持在使用Coreboot的设备上引导；支持Windows的Ntldr/bootmgr、PXE链式引导和Darwin 11（OS X 10.7）的引导等。

TeX Live 2012发布

TeX Live是一个由常用TeX程序、宏包和字体组成的排版套件，它是Fedora、Debian、Ubuntu、Gentoo、FreeBSD的默认TeX系统。

除了可以直接利用TeX语言写作文档或幻灯片外，对于Markdown、reStructuredText、Org-Mode格式文档，还可以将它作为后端排版系统，输出精致的PDF文档。

TeX Live 2012的改进主要包括：在write18限制执行模式下会默认唤起MetaPost程序；pdfTeX和Dvips的输出文件大小不再有2GB限制；Dvips将默认嵌入35个PostScript字体以保证跨平台显示的一致性；增加了对使用ARM和MIPS架构Linux平台的支持，移除对SPARC Linux和NetBSD/i386的支持。

GitHub Android

代码托管仓库GitHub发布了官方Android客户端，手机和平板用户可以从Google Play下载。它的出现，将方便更多用户融入GitHub社区。GitHub Android本身也是开源软件，其源代码可以从GitHub浏览。此前，GitHub已经推出了Windows和Mac版客户端。

通过GitHub Android，移动终端用户可以快速访问并查看所有代码库的控制面板；获取代码库的最新动态；快速响应其他使用者报告的问题。该客户端也使用了多个开源Android项目源代码，如ActionBarSherlock、Android-ViewPagerIndicator、maven-android-plugin和SyntaxHighlighter。

Objective-C新方言Eero

Eero是与Objective-C二进制兼容的一种新方言，它可以通过修改版的LLVM/clang进行编译。其目的是为了让代码编写更加流畅，提升代码的可读性和安全性。Eero的语法受到了Smalltalk、Python和Ruby的启发。Eero的设计还受到了芬兰裔美国建筑师Eero Saarinen灵活简约风格的影响。

Eero与Objective-C在语法上的明显区别包括：使用类似Python的代码缩进分隔块作用域；一般情况下，标志着语句结束分号可以省略；条件语句不需要圆括号；消息传递不需要方括号。

Eero的设计者Andy Arvanitis曾在诺基亚、德州仪器、波音等公司工作。

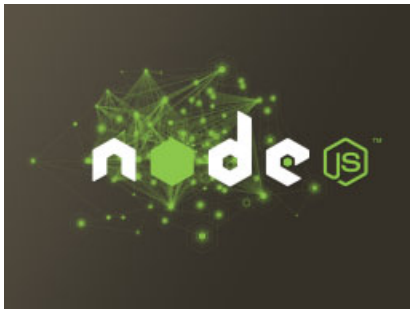
```
int main()
  helper := FileHelper new  // declare variable "helper" via type inference

  files := []  // empty array literal implies mutable
  files addObject: (helper openFile: 'readme.txt')  // can group message in parens

  for FileHandle handle in files  // all objects are pointers, so no '*' needed
    Log( 'File descriptor is %@', (NumberI(handle fileDescriptor) )
    handle closeFile

  return 0  // semicolons are optional almost everywhere

interface FileHelper : Object  // "Object" is the same as "NSObject"

  property (readonly)
    String volumeName    // both are readonly properties
    String volumeFormat  //

  + pathStringWithComponents: Array, return String  // class method

  openFile: String, (withPermissions: String), return FileHandle  // instance method

end
```

易用的Mac数据库Postgres.app

尽管Mac用户可以通过Homebrew或Ports等方式安装PostgreSQL，但为了安装包管理工具，首先需要安装Xcode命令行工具，配置和管理PostgreSQL对于许多人来说也较为烦琐。

Postgres.app是Heroku工程师Mattt Thompson开发的OS X平台PostgreSQL独立应用程序，它让PostgreSQL的易用性又提升了一步。不需要编译安装，打开Postgres.app，你就拥有了一个等待连接的PostgreSQL服务器；关闭程序，服务器便自动停止运行。

新稳定版Node 0.8发布，性能大幅提升



新的Node 0.8版速度更快、更稳定，Cluster模块有了大幅度改进，新增了Domain模块和基于GYP的新版构建系统。

从Node团队分享的性能基础测试来看，Node 0.8的性能比上一个稳定版（0.6版）提升了近200%。速度提升主要体现在吞吐量和I/O操作。Node团队指出，大部分的性能提升归功于V8的改善。V8团队对于Node.js项目响应非常积极。

新版本在Cluster模块中添加了一些新特性，并让已有特性变得更加快速和稳定。部分特性与文件描述符相关。新的Domain模块可以聚合多个I/O特性，并且更有效地处理崩溃。

Node还添加了一个新的构建系统——GYP。它可根据目标生成Makefile、Visual Studio项目文件或XCode文件，最近它也被V8和Chrome采纳。

Apache Geronimo 3.0，支持 Java 7

APACHE GERONIMO

Apache Geronimo是Apache软件基金会的开源J2EE应用服务器，它集成了许多新技术和设计理念。这些技术和理念大多源自独立的项目，配置和部署模型也各不相同。Geronimo能将这些项目和方法的配置及部署完全整合到一个统一的模型中。通过Apache Geronimo，开发者可以创建有良好扩展性和移植性的企业应用。

Apache Geronimo日前发布了3.0正式版，除了通过Java EE 6认证外，该新版本还增加了对Java 7的支持。新组件还包括：Tomcat升级到7.0.27、OSGi升级到4.3、BVal升级到0.4、Derby升级到10.8.2.2。

在线移动应用原型设计工具Fluid UI

Fluid UI是一款针对iPhone和Android的智能手机UI原型在线设计工具。它通过HTML5技术，帮助开发者完成动态原型设计。Fluid UI目前支持Safari和Chrome浏览器，对Firefox的支持也正在开发中。除了可以在桌面浏览器上运行，原型也可以运行在iOS或Android系统中。它可以模拟各种手势操作，为使用者带来更真实的操作体验。

除了具备iOS和Android系统中的上千种UI元素，它还支持翻页、淡出等动态效果。Fluid UI支持分享和协作功能，完成后的设计可以导出为PDF、HTML等格式文档。

所有“云”方案的超集——Oracle的云战略

行业内陆续出现的竞争对手似乎已在和Larry Ellision进行“非对称”的市场竞争，对于Oracle而言也不得不面临转型的关键选择。

这两年，圈子里面最不缺的就是贴着“云”标签的各家厂商、各种方案、各式各样的产品，甚至一些云服务厂商相互之间的争论都集中在谁是“新瓶装旧酒”、谁是真的“创新”上。不过听到6月6日Oracle CEO Larry Ellision在加利福尼亚的一段讲话后，估计你会认为之前所有的争论也好、讨论也罢似乎都只是“两小儿辩日”，他强调了未来7年Oracle面向云平台的一系列先手布局，包括但不限于：

- 工程、软硬件技术、技术管理手段的创新；
- 一系列战略性的收购；
- 数十亿美元的大手笔投入。

Larry Ellision眼中的Oracle云战略是地球上最全面、最专业的。不仅如此，它全部是基于标准技术完成的。比较而言，其他云方案供应商的主要缺点在于扩展性远不足，而这正是Oracle能做到的。从具体组成看，Oracle的云战略包括三部分：Oracle云平台服务、Oracle云应用服务和Oracle云社交服务；从Oracle公司的角度看，它对于三个部分技术的掌控看是逐步递进的。

- Oracle云平台服务是基于Oracle数据库、Java EE技术。通过Oracle Exadata和Oracle Exalogic集成后，集中了Oracle的“看家本领”，目的是向上形成一个基于WebLogic中间件和Web Service为接口的分布式云计算容器。区别于其他大部分厂商单一面向计算的云平台，Oracle的云平台服务是一个Application Ready的环境，也就是说标准的Java EE第三方应用及PHP、Ruby、Python程序可以通过符合传统应用标准的WebLogic Server和Web Service实现应用逻辑。另外，面向今年夏天注定“热度”难减的移动应用而言，也可以在“云平台服务”一层解决。

- 而Oracle云应用服务则是Oracle传统的商业套件，这部分可以视为Oracle在云平台服务上面向典型商业场景自己开发的通用套件。虽然Oracle在这个领域确实已经耕耘了不少年，但毕竟没有达到“一家独大”的局面。当然，以前的游戏规则是在企业应用条件下确定的，如果Oracle将这些商用套件放到云上，就诞生了一个全新的格局，尤其对于现金流越来越重要的中小企业而言，Oracle云应用服务不能不说是一个比开源还要高明的商业选择。

- 最后，在云社交服务上，Oracle力图将自己在商务智能和企业内部消息集成的优势整合在一起。作为该领域的后来者，Oracle清楚，直接提供互联网层面的社交网络服务相当于在拿投资打水漂。面临强大的竞争对手，这么做非常不经济。既然不能用自己的短处去碰竞争对手的长处，还不如换个思路，为Oracle最擅长的企业应用赋予具有安全性、可管理性的社交服务功能，便于当前大部分企业中越来越普遍的员工异地办公、移动办公需要。同时，通过Oracle擅长的数据分析技术，便于用户在使用中发现商机、找准运营问题。

这样看，Larry Ellision所谓的“最全面”其实是存在局限的，但又是“投入了数十亿美元”用以转变既有产品线。随着基于Internet的社交网络和Big Data的日趋普遍，Oracle尽管可以不断巩固在既有领域的市场优势，但行业内陆续出现的竞争对手似乎已在和它进行“非对称”的市场竞争。对于Oracle而言，也不得不面临转型的关键选择，凭借手中掌握的Oracle、Java EE和WebLogic三张牌，Oracle必须尽快把它们发布出去，确保用户迁移到另外的新平台之前，可以先把用户稳住，通过Oracle云战略这张“大饼”，吸引足量的用户人气。同时，用户可以按照Oracle的市场定位，自下而上地逐步融入三类服务之中。

Oracle的云从远方逐步飘近，云服务市场的空气中似乎酝酿着一场透雨。

王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

逐鹿反 APT

你可以根据过去预测未来。敌人不可能改变所有的手段，只要你了解了他们，就能更好地面对他们的下一波攻击。

本月一些安全会议陆续召开，包括“构建安全、和谐的网络环境”的中国计算机网络安全年会、《信息安全与通讯保密》杂志社理事会议等，这些会议的共同热点之一，是如何对抗APT（高可持续性威胁）攻击。

而与此同时，关于APT的各种最新消息，也在网络上传递交汇。

7月1日，Kaspersky宣布发现OS X上一个名为MaControl的后门软件，正在通过电子邮件附件传播蔓延。Kaspersky认为该恶意代码是APT攻击的一部分。18日，F-Secure则展示了一批文档，包括MS-Word、MS-Excel、PDF等格式，均被用于APT攻击，其中包含了漏洞利用代码，可以加载要目标计算机中。

从业界来看，新一轮产品竞赛即将浮出水面，国内企业虽然落后了两年，但跟进的速度并不慢。

对抗APT有多种技术思路，一种是演化出类似Fireeye、Damballa等反病毒领域的深度分支产品，这些产品基本上是采用流量获取+虚拟加载的方法，其重点在于应对格式溢出等，特别是针对0day。国内有瀚海源、安天等几家公司进入了这个领域。

有趣的是，我们很难从网站上和其他公开渠道中了解到这些产品的端倪。显然，反病毒的最大软肋，就在于它是一种易于获得的资源，非常容易演化为敌对体系的对抗测试Q&A环节——病毒编写者一直测试到反病毒软件失效为止。因此，反APT产品对细节讳莫如深也不难理解。

而另一种形态则演化为以白名单思想为基础、以私有云为依托的解决方案。国内有金山、江民等厂商在做类似的尝试。这种解决方案通过安全基线的方式解决执行准入问题，它是基于传统反病毒技术基础，但又有所突破的另一种尝试。

当然，私有云的问题可能与可信计算比较类似，解决的多是可执行文件的安全问题，而应对非可执行文件的威胁能力不足。同时，其鉴定效果与采集能力有关，会遇到Rootkit的挑战。

传统安全设备厂商自然也不会放弃努力，他们把很多工作放在了对长期数据的缓存和对大数据的回溯分析之上。因为今天的0day会在未来某一天变成可检测的已知漏洞。那么，既然APT将实时对抗几乎变成不可完成的任务，那么能够回溯总比一无所知更好。我预计，对这种思路更欢欣鼓舞的还是那些存储厂商。

SANS安全研究机构专家Rob Lee则从用户素质的角度看待问题。他认为，APT攻击可以被阻止，但这需要企业接受一系列训练。他指出，财富500强中已经有超过50%的企业遭到过APT攻击，“你可以根据过去预测未来。敌人不可能改变所有的手段，只要你了解了他们，就能更好地面对下一波攻击。”

APT背景下，“一切均不可靠”将成为定律。近日，一块特殊的渗透设备就被曝光。表面上它是一只插线板，但实际却是内置了Wi-Fi、以太网、蓝牙和3G通信模块的专用渗透设备，能成为入侵内网的跳板。

尽管这个方法此前曾多次见诸网络文献，但一直还停留在民间有限的尝试。而这款设备则工艺水准很高，并以1295美元的价格出售，但这些都不是最值得关注的，最引人瞩目的是该项目的背景——由DARPA资助开发。“管中窥豹，可见一斑”我们已经可以看到“大玩家”（国家与政府）入场后，网络世界的秩序图景将会有怎样的变局。

肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。微博：weibo.com/seak。

新授权协议提升 OpenStreetMap应用价值

新授权协议下的OpenStreetMap项目，将在移动位置服务的热潮下，为地图数据用户带来更好的体验，同时也为应用开发者创造更大的价值。

致力于分享全球范围免费地图数据的OpenStreetMap项目近期消息不断。OpenStreetMap本月宣布，项目将正式采取Open Database Licence（ODbL）许可向地图数据使用者授权网站免费发布的OSM数据，包括点、线、面的坐标数据，地理位置间的相互拓扑关系，以及世界各地志愿者贡献的GPX轨迹数据。

此前，OpenStreetMap一直使用CC-BY-SA协议对OSM地图数据的采集和发布进行授权。但是被个人博客站点广泛采用的CC协议并非针对数据共享，特别是地图数据共享而设计。CC协议的设计初衷是用于创作内容的版权保护，主要包含文档、照片和绘图等互联网创作形式的版权保护，而协议本身对于互联网数据的采集和共享没有清晰权责界定。并且CC协议的创建者也不推荐将此协议用于信息数据库共享，包括教育或科研等用途的数据库共享。

ODbL协议是OpenStreetMap基金会与Open Knowledge Foundation联合设计的数据共享授权协议。自2010年5月12日开始，OpenStreetMap用了两年的时间对OSM数据库中的地理位置数据进行了清理，对于那些不愿意将自己贡献的数据由CC协议转换为ODbL协议的用户，OpenStreetMap将不会在数据库中保留这些数据，而新加入的数据则默认为贡献者同意使用ODbL协议对数据的分发和使用进行授权。

OpenStreetMap在先前使用CC协议的过程中，难以对OSM地图数据的使用范围进行清晰的表述，比方说地图数据的出版发行和地图数据的商业应用。在数据共享方面，CC协议对哪些事情能做，哪些不能做没有清晰的界定，从一定层面上造成了OSM数据的滥用。对于地图API开发人员，在如何正确地将OSM数据与其他来源的地图数据混合使用，CC协议也带来了一定困扰。相比较起来，ODbL在数据授权使用上较之CC协议更为详尽和明确，除了包含CC协议在内容版权方面的保护，ODbL还在数据版权、数据库归属权利和使用范围上进行了限定，比CC协议的适应性和可操作性更强。

成立于2004年8月的OpenStreetMap项目，在创始人Steve Coast的推动下，目前已成为全球最大的地理数据共享网站，拥有超过50万个注册用户和众多活跃的地图数据贡献者。为保证项目的可持续发展，OpenStreetMap在2006年8月注册成立了基金会，来负责项目数据库服务器的管理和维护，并保证数据的合理使用。

随着OpenStreetMap项目影响力的逐步扩大和LBS移动位置应用的火热，创始人Steve Coast也被微软看中，聘为微软Bing的地图服务Bing Maps首席架构师。显然，微软希望通过开源社区的影响力，来对抗地图服务领域一直处于领头地位的Google Maps。在Steve Coast加入微软后，OpenStreetMap与微软随之在地图数据层面展开了不少合作，如在Bing Maps中增加了OSM地图图层。更值得一提的是，近期Bing Maps进行了有史以来最大规模的卫星影像数据更新。Bing Maps更新了总计165T的卫星影像和航拍正射影像，而此前Bing Maps数年积累的遥感数据总量为129T，并且Bing Maps中所有更新的卫星影像数据，都可以在OpenStreetMap中访问。

致力于创建和提供免费地图和地理信息数据的OpenStreetMap，一直以来以类似于维基百科的方式接受志愿者贡献的地理数据，并向各种地图数据的用户提供数据下载和API调用服务。在这样的业务模式下，遵循良好设计的数据授权协议对整个项目的健康发展尤为重要。在OpenStreetMap地图数据授权协议变更后不久，FourSquare随即宣布将放弃Google Maps，转而使用OpenStreetMap提供的地图数据。在新授权协议下的OpenStreetMap项目，将在移动位置服务的热潮下，为地图数据用户带来更好的体验，同时也为应用开发者创造更大的价值。

高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

艾伦·图灵
——如谜的解谜者

文 / 苏椰

2012年6月29日，是英国数学家艾伦·图灵100周年诞辰。他24岁发明图灵机模型，奠定了现代计算机的理论基础，被誉为计算机科学之父。二战期间，图灵秘密地作为英国情报界的核心人物，破译了德军的谜机密码，扭转了整个大西洋战局。战后，图灵提出了“机器能思考吗”的哲学思辨，先驱性地开创了人工智能的先河。但不幸的是，图灵因为同性恋身份，遭到迫害，以致被化学阉割。1954年，图灵中毒身亡，一代科学大师陨落，年仅42岁。本期专题中，我们谈谈通用机器、破译谜机、人工智能和毒杀之谜四个问题，缅怀这位为人类做出巨大贡献的天才人物。

通用机器

1900年，希尔伯特对数学界提出了23个未解问题。其中第二个问题，包括了事关整个数学基础的三个小问题：数学是完备的吗？数学是相容的吗？数学是可判定的吗？很快，年轻的捷克数学家哥德尔，就证明了前两个问题的答案为“否”，而第三个问题仍然悬而未决。这个问题是说，是否存在一个通用的、机械的方法，能够判定所有数学命题的真假？

1935年初夏，年轻的图灵习惯在午后沿着康河长跑，然后躺在格兰彻斯特的草地上休息，他就在这里，想到了如何回答这个问题。首先，他设计了一种假想的机器。这种机器有一条无限长的纸带，和一个可以沿纸带移动的读写头。纸带划分成无数个格子，每个格子可以是空白的，或是记录一个符号。读写头可以读取当前格子上的符号，也可以向当前格子写入符号。而最关键的是，根据当前格子上的符号，机器可以自动地切换到不同的状态，而每个状态都对应着不同的一系列操作。这样的机器结构，就是后来的有限状态自动机。

图灵证明，任何机械过程，都可以通过一张状态行为表，表示为图灵机的一个程序。也就是说，图灵机可以实现所有的机械过程，如果所有的数学命题都可以由机械过程进行判定，那么也就是说，所有的数学命题都可以由图灵机来判定。那么如果存在一个图灵机无法判定的问题，就说明不存在这样的通用的判定方法，也就回答了希尔伯特的问题。

图灵非常巧妙地找到了这样的问题：图灵机无法判定一个程序是否会终止。1936年，图灵发表了经典的数学论文《论可计算数及其在判定问题上的应用》，证否了数学的可判定性，并在这篇论文的一个脚注中，详细地描述了图灵机。图灵机只是一个用来研究数学的辅助模型，并不是一台真正的机器，然而在三年之后，一个事件使图灵机真正地来到了世界。这个事件成就了图灵，却带来了数百万生灵涂炭。

破译谜机

这个事件，就是第二次世界大战。1939年9月4日，图灵前往布莱切利庄园报到。这个坐落在山谷中的静谧的庄园，此时还有一个特殊的名称：政府编

码与密码学校。在整个第二次世界大战过程中，这里是英国的情报核心，负责截获轴心国的军事通信，并破译出军事情报。图灵作为剑桥大学的年轻数学家，是第一批被征召到这里的专家之一。

德国的官方通信，全面部署了一种名为“谜机（Enigma）”的密码设备。这种设备有三个轮盘，每个轮盘可以对26个字母进行一次映射，也就是说，每个字母都会被3次映射加密三次。更厉害的是，每输入一个字母，轮盘就会自动地转动一格，也就是说，在一条信息中，每一个字符都是用不同的密钥进行加密的。这就导致当时一般的破译方法根本不可能破译这种加密，德国方面因此对军事通信抱以极大的信心。事实上，直到德国投降时，德国都不知道，他们号称“绝不可能”的事情，早在1940年，就被一位年轻的数学家做到了。

图灵在布莱切利，利用他天才般的数学头脑，破译了看似不可能的谜机。详细的过程非常复杂，篇幅所限，我们无法在这篇文章中叙述。对此感兴趣的读者，可以查阅湖南科技出版社出版的《艾伦·图灵传》一书。图灵作为一个对政治最不感兴趣的学者，从事数学研究正是为了逃避现实社会的政治，但造化弄人，他却偏偏因为数学研究，被卷入了世界政治旋涡的正中心。然而，图灵却在这个本不属于他的位置上，发挥了巨大的作用。在图灵破译了谜机之后，德军潜艇每天的行动计划尽为盟军所知。1940年4月，英国海运物资的损失量是70万吨，到了这一年12月，德国的潜艇行动量是4月的两倍，但英国的损失却降到10万吨。作为一个岛国，海上运输是英国赖以为生的根本，如果没有图灵，二战的局势将会完全不同，世界的历史也将有可能被改写。

人工智能

二战结束后，图灵又先驱性地开创了一个全新的研究领域——人工智能。其实早在二战前，年轻的图灵就开始思考，到底什么是“思考”，机器是否能够思考。他不是一个坚定的有神论者，但也不是一个坚定的决定论者，他的思想当中存在许多类似的矛盾，这些矛盾反而使他更为客观地看待智能问题。1950年10月，他发表了一篇题为《计算机器与智能》的论文，这篇文章集中地提出了人工智能这个概念，开创了人工智能这个带有科幻色彩的新学科。也正是在这篇文章中，图灵提出了后来被称为“图灵测试”的实验方法，以此回避与智能有关的哲学困境。在图灵看来，如果一台机器的行为，让人类无法辨别它是机器还是人类，那么就可以认为，这台机器具有了人类智能。这种只关心外在行为，不关心内在机制的观点，后来形成了一个学派，被称为“行为主义人工智能”，图灵本人自然成了这个学派的代表人物。他认为，即使是人，也无法真正地判断其他人是否具有“思维”，他只能将其与自己进行比较，因此，人类没有任何理由不以同样的原则来对待机器。尽管图灵胸有成竹，但曼彻斯特计算机的性能，远远不够把他的想法变成现实，事实上，当时世界上任何一台计算机都不可能做到。图灵面临的问题，是深远的洞察力与当时技术水平的严重脱节，但幸运的是，这篇论文在被埋没之前，已把最原始的强烈愿望，传达给了整个世界。尤其是图灵的行为主义原则，在现在的人工智能技术中，已占据了绝对的主流。计算机这门科学，很大程度上体现了几位关键人物的个性。图灵在战争期间，是作为一个隐秘的情报破译者，没有人知道他藏在哪里，也没有人知道他是如何工作的，人们只关心他破译出来的结果。他就是当时的英国的“大脑”，他也认为“大脑”本该是这个样子，人工的“大脑”也应该是这个样子——与外界的交互仅仅依靠字符就足够，而且外界不必理解其内部机制。可以说，他的个人经历，对这个学科产生了极为深远的影响。但在当时，可以想见的是，这样超越时代的想法，会遇到巨大的阻力。图灵的“人工智能”，遭到了来自科学、工程、哲学、社会、宗教等各个方面的猛烈攻击，但图灵踌躇满志地说：我相信，在50年之后，一定会实现这样的智能机器，可以用自然语言与人类聊天，而且让人类在短时间内无法发现它是机器。机器能思考吗，这个问题，会自然地失去意义，根本不值得再讨论。

毒杀之谜

就在他准备施展拳脚时，灾难却突然降临。图灵

不但在学术上非常前卫，在生活中也有一个超越时代的秘密：他是个同性恋者。1952年，图灵的住处失窃，在报案过程中，他与男伴同居的事实被告发。图灵被逮捕了。在法庭上，图灵坚决声称同性恋无罪，结果可以想见，他被判有罪。他只有两条路，要么入狱，要么化学阉割，他选择了后者。1954年6月8日清晨，女管家发现图灵的心跳停止了，床头放着一只咬了一口的苹果，上面沾有图灵亲手提炼的高纯度氰化物。他究竟为何而死，是意外，是自杀，还是隐藏着更大的阴谋？这就不得而知了。有人认为，阉割给图灵带来了不堪忍受的屈辱，进而导致他选择轻生。但还有证据表明，图灵提炼氰化钾，是为给一块手表镀金，那么是否有可能是在提炼过程中，不慎沾到了苹果上而不小心误食？图灵在逝世的前两天，还预约了下个星期的计算机使用权，并且编好了程序准备做实验，这完全不像是自杀之前的行为。但如果说不是自杀，他又为什么在逝世前留下了遗书？这位如谜的解谜大师，最终给世人留下了一个永远解不开的谜。

开放中创新

体验Google I/O 2012

Google I/O讨论的焦点是使用Google和开放网络技术开发网络应用。《程序员》杂志邀请了4位亲历Google I/O 2012大会的业内人士，从不同角度分享自己的亲身感受。

杨武
上海改变科技有限公司创始人、总经理，开发过keyOne、AirSlides、RockPlayer等产品，iCosta输入法共同作者。

成功的偶然与必然

Google在会上发布了Nexus Galaxy手机、Nexus 7平板、Nexus Q媒体播放器、Chrome Box迷你电脑主机等硬件产品，并且给参会的开发者每人发了一套，对开发者很是慷慨。不过Nexus Q立即让我想起了Apple TV，而Chrome Box则实在太接近Mac Mini了。

回想起6月去GooglePlex参观，正好有个朋友在Nexus Q团队工作，告诉我有个发给参会者的神秘产品，正在努力地消灭Bug，赶在会前稳定下来。看来产品赶在发行前几天变得能见人，大小公司都一样。

Google同时还发布了多项软件和服务更新，最引人注目的是，Google Play Store除了应用，还可以销售音乐、电影电视、杂志书籍。iTunes有的Google Play Store都要有？Android新版本Jelly Bean 4.1通知栏的进一步改进是为数不多的iOS需要学习的地方，而语音助理则完全是在向Siri致敬了。试用下来，语音识别方面Google胜出。而语义理解和协助方面，如果Siri是个玩具的话，Google Voice Assistant只能算个原型。从Jelly Bean开始，Chrome是系统默认浏览器了，Chrome Mobile的Tab管理很有创意，允许用户打开很多标签页，在标签页头上滑动切换确实很方便。

抛开业内人士的批判眼光，这些设备和服务对于用户来说是件大好事，因为能更从多渠道获取内容。不过Nexus Q只能播放来自Google Play Store音乐，是不是有点儿过头了？即使iTunes也能导入CD然后流化到Apple TV上的嘛。

当第一天的主题演讲到了Google+环节时，更觉无趣。之前已经听过Google朋友抱怨，他们对于Google+拿出来的数据很不满，大概是因为通过统一Google账号，很多并不使用Google+的账号也被算进去了，而且通过Android设备激活过程来烦用户绑定Google+就已经够可以的了，Google+团队还在试图让YouTube账号绑定Google+。

说到Google Glass，这次遇到一位印度朋友，他曾经师从可穿戴计算机的先锋Thad Starner。Google Glass项目就起源于他的研究工作，目前他在眼镜项目做Tech Leader。

很久以来，都有一种感觉，Google在正确的时候把搜索做得很出色，于是迅速成长起来了，纯属偶然，此后对于如何从搜索巨头更进一步却没什么方向。而Google Glass项目及发布方式仿佛在说，Google年轻又有活力，没准又能抓住下一个大机会。保持玩心，让一切变得更酷更好玩，或许是互联网走向巨大成功的必由之路吧。

王佳梁
触宝科技联合创始人兼CEO。毕业于上海交通大学电子工程系，曾任微软研发集团项目经理。

感受颠覆式创新

Google Glass

从个人角度来说，Google Glass是我这次最期待的产品。但对我而言，不远万里来到美国，不仅是为了那些数码产品，更是为了感受颠覆式创新带给我的兴奋，为了现场感受技术宅男们那种充满激情的狂欢。

因此，当Sergey打断了演讲，戴着眼镜冲上舞台的那一瞬，我激动得从座位上跳起来欢呼也就不足为奇了。当然，我并不用感到尴尬，因为身边参会者似乎比我更为激动。那些戴着眼镜，前一秒还斯文地摆弄着手机或敲打着笔记本键盘，看上去有点儿像生活大爆炸中Sheldon的Geek们，此刻爆发出的能量，足以让摇滚乐队现场的狂热歌迷们相形见绌。

这仅仅是开始，后面的情节更加疯狂而不可思议：几位跳伞高手戴着Google Glass从热气球上跳下，而第一视角通过Glass的视频拍摄投放到了现场的大屏幕上。更酷的是，那些跳伞者正好落到了会场顶楼，并进入了大会现场。当前一分钟还在屏幕上看到的跳伞者，此刻从你的身边走过时，那些在普通人眼中不切实际、遥不可及的梦想，也许离我们只有咫尺之遥。

我突然明白为什么美国人可以创造出iPhone，创造出Facebook，创造出一个又一个的科技神话。并不是因为他们的技术更强，而是因为他们拥有更加疯狂的梦想，以及将这些梦想变成现实的勇气！

会后，我也有机会近距离接触了Google Glass。虽然不能亲身试戴，但我还是很期待自己预定的Google Glass在明年初能按时送到。

Google No

另一个让我眼前一亮的功能是Google Now。我一直觉得，手机最终应该成为生活助理，它不仅能查询信息，更可以在你需要的时候自动向你推送信息。尽管目前Google Now可以做的还比较少，但这个方向我非常认同。

大会结束后的几天，我们一行人相约去斯坦福大学参观。参观结束，大家都想去买一些纪念品。斯坦福的朋友建议去书店，但不知道书店是否已经关门。于是我拿起手机，很自然地打算搜一下书店的营业时间和地址。没想到Google Now很神奇地在搜索栏下方显示出Stanford Book Store的位置和营业时间。

另一个很方便的用途是查询路线。有一次在旧金山开完会，我还要赶去Mountain View开下一个会议。我提前就把开会的时间和地点记录到了手机的日历中。而此时，Google Now中就显示出到下一个会议地点的路程，以及所需的时间。

我隐隐感觉，Google Now将是一个具有划时代意义的产品。它将改变人们搜索的方式，是移动搜索的未来趋势。

钟文昌
索尼移动通信架构与管理部架构师。

从Google I/O 2012看Android 4.1的变化

Android 4.1 UI的改进

Google在I/O大会上发布了Android SDK 4.1，乍看之下与4.0并无差异，但从开发者的角度，就能发现其UI和底层都有相当大的改进，尤其是在显示和性能方面。Android 4.1在画面显示的部分通过VSync及Triple Buffering加速影像同步，使成像、卷动、翻页及动画部分更为流畅。

从Android 4.1中可以看到的改进还包括：

- 增加了对更多国家、语系的支持，支持Bi-Directional，能够依据不同语系选择从左到右或从右到左的输入方向。
- 支持可延展的Notification，依据当下所需，动态调整Notification的大小，能够提供大型且丰富的Notification资讯。
- 支持720×720的联络人照片。

Android 4.1底层改进

Android Beam是一项以NFC为基础的传输技术，能够传输文档、照片、影片等。

Android 4.1增加了对硬件装置变动的支持，在输入模块部份，上层应用能够向系统注册以接收底层硬件变动的通知。当底层硬件连接发生变动时，上层能够即时收到通知，进而采取相应的处理。

Wi-Fi无疑是目前最理想的无线传输方式，而Wi-Fi Direct能够提供高速点对点数据传输。除了Wi-Fi Direct，Android 4.1还增加对其API的支持，让具备Wi-Fi装置的使用者能够相互侦测并且直接配对。

Android 4.1支持AAC 5.1声道编解码、Multichannel Audio，还支持HDMI和USB Audio输出，例如在Audio Docks上输出声音。这项功能也随着Open Accessory Development Kit公开。

Open Accessory Development Kit是今年我最感兴趣的主题，因为随着开源硬件和软件的概念逐渐被人们接受，现在的嵌入式系统已与以往大不相同，发展也非常迅猛。

HTML5大行其道，Android 4.1在浏览器以及WebView部分做了性能改进，例如加快画面显示速度，减少内存使用量，改善画面卷动及缩放效果。系统采用了更快的JavaScript引擎，支持HTML5的富媒体功能，让用户能够得到更好的影音体验。

工具部分，Android 4.1 SDK中提供一个新的名为systrace的工具，能够从Kernel到系统层对App进行性能优化。此外，还为开发者提供了多种除错及调校功能，例如视窗图层更新、GPU图形处理等。

App更新也更加智能，可以仅更新APK有修改的部分，而非像以往那样必须完整下载新版应用程序，如此一来，能够节省2/3的下载流量。

总结

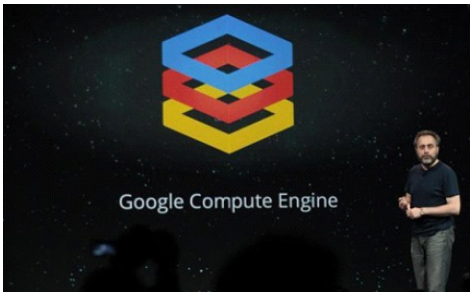
目前全球手机年出货量超过16亿，其中智能手机只占三成，上涨空间巨大。各家厂商也无不使尽全力争抢这块大饼。但就纯硬件而言，利润微乎其微，整个市场已倾向大者恒大，很少有后进者愿意加入这个血流成河、毛利却不到5%的市场。于是，软硬整合，以及软件市场是当前竞争的焦点。目前嵌入式系统开发模式已PC化，软硬整合的关键在于如何创造出产品的差异，而至于纯软件部分，强调的是创意及本地化服务。

构建完整的云计算体系

陈世欣

1999年毕业于上海交通大学计算机系，获硕士学位。之后一直从事国内外互联网创业公司的技术和产品相关工作。现在是欧洲2Style4You公司CTO。

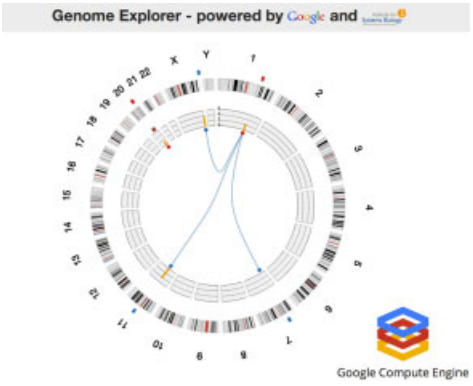
在本次Google I/O大会上，Google宣布推出了面向虚拟基础设施的云计算服务：Google Compute Engine，它可以为客户提供Linux虚拟云服务器。



我预计，Google将从Amazon手里抢走一批坚定的云服务客户，因为Google的云计算平台有更成熟的数据中心，性能更有优势、价格更低廉。

从规模来说，由于Google内部早就在使用云计算的方式处理大量的日常计算任务——每天需要处理数十亿搜的索查询。Google服务器早已超过100万台，其规模之大，无与伦比。如今，任何客户都可以利用Google Compute Engine，这将大大减少他们管理任务的时间，简化计算流程。

Google展示了一个癌症研究工具，它使用了超过60万个CPU内核。



Google数据中心的效率和稳定性足以保证云端服务的连续性，不会发生突然掉线的情况，用户可以随意调整应用程序规模而不用担心服务出现问题。

从价格来说，Google数据中心所提供服务的性价比很高。与其他云端服务商相比，Google Compute Engine在同样的价格上可以多提供50%的计算量。

从性能来说，技术力量雄厚的Google也会是赢家，如果Google能够解决Amazon用户经常遇到的I/O性能、网络性能问题，Google将迅速获得大量Amazon的客户。

Google Compute Cloud的发布意味着Google在云计算服务端的三个层面上布局全部完成。

SaaS层面：Google Docs、Google Drive、GMail

PaaS层面：Google App Engine、Google Cloud Storage、Google BigQuery、Google Prediction API、Google Translation API

IaaS层面：Google Compute Engine

除了服务器端布局之外，Google在客户端的布局也已全部完成：

浏览器端有Chrome浏览器；在操作系统端有ChromeOS；在PC硬件终端有Chrome Book和Chrome Box这样完全使用云计算的硬件设备；在移动硬件终端，有Android手机、平板、Android Play、Google TV和Google Glass。

可以想象，未来的应用可能完全基于Google云计算体系而构建，而客户端方面则是多种多样的。

中国互联网企业的研发之路
与腾讯研究院院长郑全战一席谈

文 / 刘江

腾讯研究院是国内互联网企业中最早的研究机构之一。2012年恰逢腾讯研究院五周年，《程序员》杂志总编刘江访问了腾讯首席架构师兼研究院院长郑全战。

在高科技产业中，中国的互联网可以算是异类。互联网虽然是欧美人的发明，但以腾讯、百度、阿里巴巴、新浪为代表的本土企业从蹒跚学步、亦步亦趋的模仿开始，在不到20年的时间里，通过扎根本土不断研发和积累，居然最终在各个领域都后来居上。而国外的竞争者要么水土不服，要么受政府管制之困，纷纷退出了主战场。

其实，中国的IT产业也曾有过惨痛的过去。二十世纪九十年代，计算机专业的毕业生如果想干技术，其实没有太多选择。除了和无线电等专业竞争进入当时并不强大的华为、中兴等电信设备厂商之外，就是去银行、电信等企业的计算中心。那是一个贸工技为主流的时代，没有互联网，软件公司都很弱小。全球IT产业的核心是操作系统和计算机芯片，都是美国人的天下。

1994年，郑全战从北大计算机研究生毕业后，也选择到美国深造攻读计算机博士，在那个时代是再自然不过也非常令人羡慕的道路。

微软与腾讯

在微软工作9年之后，郑全战在2006年选择加盟腾讯，这出乎不少人的意料。那时候腾讯并不被人看好，商务和专业人士普遍觉得QQ只是小孩子的玩具。郑全战当时的理由也很简单：腾讯具备潜力。在微软工作的经历培养了他的远见："用户数量是关键的因素，如果用户数量少而且没有成长性，公司是很难做大的。"而腾讯恰恰具备用户数量已很大且仍然在高速成长这两点。

郑全战很快体会到微软和腾讯的显著不同。微软是大团队作战（1997年Windows 98开发团队有四千人），成员之间的分工细致，流程规范成熟。而腾讯当时的研发流程和代码管理都很不规范，但决策很灵活，团队小而精干，产品研发效率极高，是典型互联网公司式的小步快跑。在谈到这些年感触最深的事时，郑全战举了下面的例子：

"我常有一些关于产品的想法，但在微软无法实现。而在腾讯，有了合适的想法，马上就可以动手做起来。我一直觉得Windows Media Player难用，所以到腾讯一有机会，我就着手重写播放器。我们当时只用两三个人的团队就完成了QQ影音。它有很多创新点，后来被马化腾做成范例，出现在他谈及怎样做产品、怎样做产品经理的PPT里。"

人才，人才！

然而，此时腾讯与微软最大的差距，是人才和技术。而核心技术的建立，归根结底，还是人才。

为了吸引高端人才，打造腾讯核心竞争力，时任公司联席CTO的熊明华开始筹划成立研究院，有着丰富研发经验的郑全战成了院长的合适人选。2007年10月15日，国内互联网企业中第一家专门研究机构——腾讯研究院在北京、上海和深圳三地同步设立。最初的成员只有30人，第一批人才大多是与郑全战熟悉，出于对他个人的信任才选择加入的。

随后，研究院一方面锻炼队伍，用产品和成果说话；另一方面也与中国顶尖高校和科研机构建立了合作伙伴关系，每年举办全国级高端技术创新大赛。在此过程中口碑效应逐渐发挥作用，有越来越多的优秀人才主动加盟。其中已成为领军人物的就有QQ旋风的黄琰、QQ影音的Simon等。

在外招人才的同时，腾讯研究院更愿意自己培养，因为这对公司的文化传承非常重要。QQ浏览器的司天歌是其中的代表。

对于人才的选择和培养，腾讯研究院看重工作热情、聪明程度、学习能力和合作性。郑全战认为，即使是一流人才，没有合作能力，也会遇到职业瓶颈。

研发之路

研发与创新并非易事，尤其是有组织地进行，而结果往往毁誉参半。历史上著名的实验室如贝尔实验室和施乐的Palo Alto研究中心都是如此。它们虽然创造出不少后来影响人类社会的成果，但对公司本身却没有起到什么关键作用。IBM、微软等公司在国内的研究院虽然经营多年，但成功转化的研究成果也并不多。

相比之下，腾讯研究院走出了一条很有特色的研发之路。

- 研究院在模式识别、多媒体通信、数据挖掘、图像处理、分词等方面取得了诸多成果。到2011年，腾讯申请专利超过4000件，超过国内其他互联网企业总和，其中研究院的贡献超过一半。与此同时，这些技术又应用于QQ、微信、搜搜等产品，有力地支持了业务发展。
- 研究院本身是孵化器。5年来开发并运营了QQ浏览器、QQ影音、QQ输入法、搜搜地图街景等许多在市场上后来居上的成功产品。
- 研究院又是人才培训基地。5年间，研究院总共培养和输出了将近千人，培养了一批公司核心骨干，其中总监级以上的就有20多名。

如果对照前不久Google阐述其混合式研究的论文不难发现，腾讯研究院与前者殊途同归。那就是模糊研究和产品开发之间的界线，同时实现学术研究成果和产品的价值。

郑全战博士（右），腾讯公司首席架构师，腾讯研究院院长。在美国微软总部工作时，全程参与了Windows 98到Vista，以及DirectX 6.0到9.0的研发。郑博士曾于1992年翻译了微软在中国正式出版的第一本书《Microsoft Windows开发环境与技术》，是最早在中国介绍Windows NT技术的人之一。郑博士于北京大学计算机系获学士和硕士学位，美国明尼苏达大学计算机系获博士学位。

面对业界关注的如何与腾讯共存的问题，郑全战表示，腾讯的重心已转为平台建设，为开发者提供便利的开发环境，因此腾讯内部对模仿的审查越来越严格，力求杜绝纯粹模仿性产品出现。

未来的挑战

近年来中国互联网的本土优势正在面临严峻的挑战。随着Apple iPhone和Google Android为代表的智能移动设备、以Amazon为代表的云计算以及大数据的兴起，整个信息产业进入技术大变革时期。中国本土的互联网公司又一次落在了后面。

而成立5年的腾讯研究院也随着整个公司的战略转型、机构改组经历了一次调整，未来将更多地专注于人机交互、多媒体、网络、移动技术等互联网前沿技术研发，为公司业务发展提供互联网基础应用组件或技术平台。这符合技术变革的大趋势。

当然，对于已成为国内互联网企业领袖之一的腾讯，对于国内首家互联网企业研究院，我们理应抱有更高期望。

做有中国特色的云计算

IBM大中华区系统与科技部电信行业及OEM业务总经理侯淼专访

记者 / 董世晓

2007年，“云计算”这个字眼开始进入中国人的视野。这五年，中国云计算得到了长足的发展，已从开始的概念炒作逐步落地。但通过与国外云计算相比较，我们不难发现，中国云计算有些与之颇为不同的特点。那么，这些特点出现的根源在哪里？如何看待这些特点？IBM作为云计算的领导企业，积极推进中国云计算事业，对中国云计算有深刻的理解和洞察，正因如此，我们近期采访了IBM大中华区系统与科技部电信行业及OEM业务总经理侯淼，请他对中国云计算的特点进行解读。

按行业划分中国云计算

众所周知，国外云计算往往是以企业为导向的，因此得以诞生Amazon云、Google云、Facebook云……但在国内，云计算却另辟蹊径，往往被命名为“行业云”。

关于这个问题，在侯淼看来，中国之所以能独树一帜推出“行业云”的概念，是由于中国云计算是与行业用户密切相关的，而不同行业对云计算的需求是不同的。例如电信行业对云计算的需求是节约成本、灵活部署，其所需要的硬件、软件都会围绕这两个需求；而零售行业或者金融行业更多的是将云计算定义为其内部的管理系统，更多强调的是高可用性。

下面以电信行业为例，分析其对云计算的两大需求。

节约成本。由于电信行业以前构建的硬件平台和基地比较多，每年都会按项目购买设备，是一种烟囱式的结构。但随着机器数量的增多，承载机器的成本，包括机房面积、电力、维护人员等，越来越高；加之电信行业近年的发展速度渐趋平缓，所承受的利润压力加大，对固定资产的投入也比较谨慎。因此，电信行业有一种动力，希望通过云计算的方式使众多小系统整合为大系统，以减少机器的物理数量，提高整个机器的使用效率，并最终降低IT投资。

灵活部署。因为电信行业以前使用IT的地方大多是内部IT，包括计费、客户关系、管理系统，都与其管理相关。但电信公司这些年在全国建了多个数据中心，其中承载了诸多新业务，例如手机互联网业务。这些业务的种类和特点，和之前相比已大不相同。因此，电信行业希望IT系统能够更加灵活，以满足未来的发展。而云计算恰好能满足电信行业的这些需求，因此被看做是一个正确的方向。

私有云在中国最易落地

国外云计算大多是基于公有云的，用户可以在一个中心购买自己需要的虚拟资源，这个中心拥有强大的能力承载大量用户。Amazon、Rackspace、Eucalyptus就是这种“中心”的典型代表。以Rackspace为例，它在全美有数十个数据中心，借助于发达的互联网网络，通过虚拟化技术，将其资源出售给用户，大幅减少了企业IT硬件方面的投入；并且，因为这些工作已由“中心”的专业人员操作，使用户无需关心运维的问题，从而减少了用户的IT人员投入。

与之相比，目前公有云在中国普及存在一定的

难度。以北美为例，由于有着相对完善的相关法律，云计算用户不必太担心数据安全。但在中国，首先，大部分国企都有自己的IT设备、机房和IT人员，因此基本上没有对公有云的需求；其次，私营企业可能会担心数据安全。

由此可以看出，目前的这些现状，使得私有云推起来会比公共云更容易找到客户的共性，引起客户的共鸣，因此决定了中国公有云发展较缓。

公有云落地尚需时日

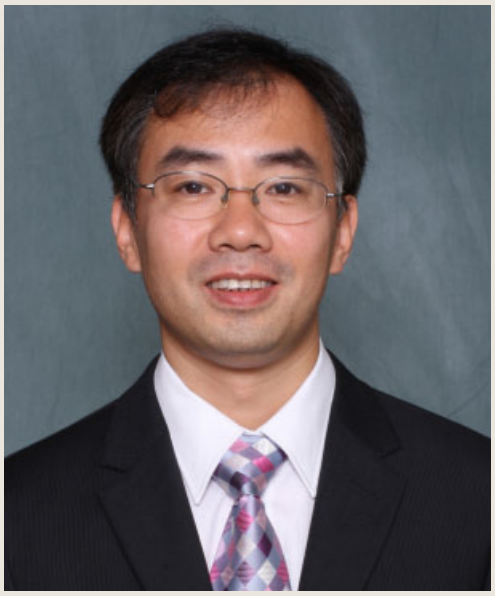
在侯淼看来，中国发展公有云，前景广阔，而且目前电力、通信等行业以及一些互联网公司，都有想法和行动在做公有云方面的努力，但还有相当长的路要走，因为还有两个问题亟待解决——第一是解决客户在哪里，第二是怎样通过相应的法律和法规，保证客户的利益，特别是数据安全。此外，一个不争的事实是，面向企业的公有云的发展显然落后于面向个人的公有云服务。

从目前看来，电信行业在公有云落地方面具备较大先天优势——它能提供从线路到应用的整体的解决方案，而且部分应用开发商已深入到企业核心业务层做深入的开发。但公有云的主导权最终花落谁家，还未可知。

侯淼为中国公有云推进最大桎梏和瓶颈的诚信问题，从两个方面提出了解决方案：一方面提高IT技术，在美国的一些数据中心，已有技术能够实现保证分区和分区、终端和终端的数据本身并不干扰，而且能保证数据端到端的安全和一致，但国内由于没有这么大的网络，很多技术还用不上；另一方面，国内对诚信、对数据安全总要一个接受和改变的过程，这估计还得过几年。

基于客户体验做云计算

据侯淼介绍，IBM在做云计算时，基本出发点在于很多关键系统都是构建在Power架构上的，Power本身是客户的生产系统，而且规模也比较大，但由于大多数系统都是按项目来的，比较分散，所以整个硬件的利用率并不是很高。但近几年，特别是Power6、Power7以后，Power在支持虚



侯淼：中国公有云落地，还有相当长的路要走

拟化、支持微分区、支持自动化节能降耗等方面的很多新的财富，为客户提供了构建更加灵活的资源池的基础。因此，围绕这个前提，IBM做了很多工作，派出其技术人员和实验室人员，跟客户一起来做云计算，并因此构建了一些很成功的资源池。通过这种方式，客户有机会是打通其多个项目平台，构建一个大的系统。未来当它上新的系统时，只需划分资源即可，而不需要采购新设备，这一方面节约了资金，另一方面节省了时间。这便是以Power为主的云计算落地所带来的一个好处，即Power产品的效率得到了提高。

此外，客户以前的系统基本上是异构的，有Unix、Windows、Linux……过去IBM做的很多工作，在于希望建立一个以Power为中心的模型，从Power产品的角度去管理不同的Unix平台、不同的x86的平台。这本身是一项长期的工作，而日前IBM推出的专家集成云平台PureSystems已为打破异构平台的藩篱，迈出了坚实的一步，实现了Scale out的高可扩展性；IBM刚刚收购了跨平台虚拟化软件大厂Platform Computing，体现了IBM除了在优化硬件本身努力之外，越来越向云计算管理方面发展。

因此，用侯淼的话总结就是：IBM所能提供的硬件平台，不单单是自己的一个产品，而且希望为客户提供一种体验，使他们能够管理不同平台，使用起来更方便、部署起来更快捷。

我们的开源

本期封面报道聚焦中国本土开发人员和团队主导和参与的开源项目，从开源项目本身、企业与个人的参与情况、开源社区建设及开源法律知识等多方面，全景式地解读中国的开源现状。

大势所趋话开源

中国开源现状分析

文 / 魏永明

随着计算机技术的发展，尤其是互联网技术和相关企业的兴起，开源软件在操作系统、编译工具链、数据库、Web服务器、移动操作系统等各方面已成为主流。而且许多企业利用开源软件形成了独特的商业模式。比如Google的Android操作系统，从2007年开源发布第一个版本起，到今天已经发展到4.1版本，占据了智能手机操作系统一半以上的市场份额，Google也通过Android操作系统在移动互联网这一新兴行业中占据了领先和主导地位。以前一直和开源软件做斗争的微软公司，为顺应潮流，也开始拥抱开源，比如向Samba项目贡献代码，放弃自己研发多年的大数据项目而选择Hadoop为其大数据的核心等。

显然，纵观IT行业这20多年的发展，开源项目已形成一股推进计算机及相关行业不停进步的巨大力量。本文将重点讲述国内的开源项目及社区的现状，以及发展过程中面临的困难和问题。

国内开源项目的发展及社区现状

国内开源软件的发展简史

国内开源项目的发展始于1997年前后。那时，中国第一个（局部）互联网（CERNET）刚建立不久，1995年在清华大学建立的著名的水木清华BBS就是开源项目。之后，Linux内核以及GNU项目中的成百上千个开源项目逐渐展现在国人的面前。

随后，中国也逐渐出现了一些开源项目。最初由国人开发的开源项目，主要解决的是Linux系统的汉化问题，流传最为广泛的应该是可以显示和输入中文的伪终端应用程序CCE。在1998年之后的两三年内，出现了以下三个开源项目。

- LVS（Linux Virtual Server）是由章文嵩博士开发的，后来被Linux内核收录，成为使用Linux操作系统搭建集群服务器的重要核心软件组件。
- Smart Boot Manager是当时的清华大学博士生苏哲开发的，它是一个引导管理器，类似现在流行的GRUB，主要解决引导多种操作系统的问题。苏哲后来主持开发的SCIM系统，被各种流行的Linux发行版收录，成为Linux操作系统上提供多语种输入法支持的标准框架。
- MiniGUI是由我开发的，后来由我创立的北京飞漫软件技术有限公司维护和发展，在功能手机、数码相框、工业控制系统和工业仪表中得到了广泛应用。

上述三个开源软件是中国开源软件早期的代表作，在国际上具有较强的影响力。

之后，国内开源项目的发展长期处于停滞状态，这与2000年左右互联网泡沫的破裂有一定关系。互联网泡沫的破裂，让许多梦想通过开源项目来创造商业奇迹的Linux发行版厂商很受伤。我曾供职的蓝点软件，在NASDAQ OTCBB板借壳上市，半年之内股价从20多美元跌到0.2美元，后于2001年贱卖。

从2005年起，开源项目的发展逐渐走出低谷。在此期间，国内也出现了为数不多的开源项目，其中以清华大学陈渝副教授主持的SkyEye最具代表性。该项目旨在提供一个面向嵌入式软件开发和调试的ARM或其他架构的纯软件仿真器（虚拟机）。该项目持续活跃长达7年时间，吸引了许多海外高手参与，是为数不多的具有国际影响力且充分体现了国际化协作、分享的开源项目。

在参与开源项目的企业当中，最为活跃的是淘宝，接下来是新浪、百度、腾讯和华为等。同时，随着

“开源中国”等社区的兴起，个人主持或者参与的开源项目逐渐多了起来。根据“开源中国”收录的开源软件，当前已经有一千多个由国人开发或者主持的开源软件。这和十年前相比，有了非常大的进步。有兴趣的读者可访问http://www.oschina.net。以下开源项目非常值得一提。

■ TFS是一款由淘宝开发的分布式对象存储系统，于2010年9月开源，存储了淘宝的几百亿张图片和交易快照。目前新浪微博已在生产系统中使用TFS作图片等对象的存储。

■ RT-Thread。这是一个由国人主持开发的开源实时操作系统，曾获得“第六届中日韩开源软件竞赛”的技术优胜奖（其他两个技术优胜奖获得者为淘宝的OceanBase和红旗的Qomo Linux）。RT-Thread 目前也获得了诸多商业应用。

■ Linux Deepin是近几年发展起来的面向桌面的中文Linux发行版，由一群来自武汉的Linux高手发起并维护。

■ ucore。从2010年暑假开始，陈渝博士组织清华大学学生开展教学用开源操作系统ucore的设计与实现，并直接用于清华大学的操作系统课程，学生可参考实验文档和ucore源码通过实践逐步深入掌握操作系统。这相对国内操作系统旧有的教学方法有较大改变，获得了国内外操作系统教学领域专家的认可，并将在教育部的支持下进行更大范围的推广。

国内开源软件的特点和问题

国内开源项目存在很多问题，如缺乏重量级项目、缺乏持续维护和更新、质量一般、用户不多等。另外，正如开源中国创始人所言，国人所开发的这些开源项目，和国际主流开源项目脱节严重，绝大多数处于单打独斗的状态。

例如，淘宝主导或参与的开源软件，大多数和互联网服务器后台、云计算相关，这些项目的主要用户是淘宝自己。因为门户之见，这些软件很难被其他的互联网企业所使用，大家不停地“造轮子”而忽视了开源软件发展必须具备的“共享”、“协作”精神。不过，现在这种情况正在改变，上面提到的淘宝TFS系统已被其他互联网企业使用，ucore项目也得到了诸多国内外大学积极的响应和支持。

我希望国内的开源项目能够和国际主流的开源项目步伐保持一致，要么加入国际化的开源软件，要么将自己主持的开源软件逐步国际化。这样，我们的开源项目才能得到源源不断的前进动力，也才能在国际化舞台上扮演更加重要的角色。

新的力量

无论如何，国内大型IT企业参与开源项目就是一个良好的开端，将为中国开源项目的发展起到非常大的促进作用。

与此同时，各种开源社区活动也越来越活跃，例如具有政府背景的“开源软件高峰论坛”和草根性质的“我们的开源项目”巡回展演等。这表明，开源软件即将在国内引起新一轮的发展浪潮。

开源我的软件？

在高物价、高房价的今天，大部分人对此问题的第一反应是：“我就是一刚解决温饱的码农，我开源，谁养我？”这几乎与我们在十年前推广开源项目理念时遇到的问题一样。但这已大大落后于时代了！我们不仅可以通过使用其他人的开源项目赚钱，还可以通过开源自己的项目来赚钱。

如何靠开源项目赚钱？

在证明上述论点之前，我们先看看别人是如何利用开源项目赚钱的。靠开源项目赚钱的方式（经过验证的）无外乎有如下几种。

■ 双许可证模式。在采取严格的开源软件许可证的同时（通常选择GPL），给商业用户提供非GPL许可方式。这本质上是一种贩卖软件许可的行为，但开源软件带给开发者一个很大的好处，即传播迅速、快速迭代。我主持的MiniGUI项目就采用这种模式，在过去的五年当中，获得了几千万元的软件许可收费。当然，使用这个模式最成功的当属MySQL。

■ 基础软件采用宽松许可证，同时向基础软件的商业用户贩卖增值服务或者增强组件、开发工具

等的许可。这种模式可用于类似RT-Thread这类的基础性软件上，RT-Thread本身可以是开源且可无偿商用的，但其上的各种增值组件，如网络、文件系统、图形系统等，可以是商业软件。国外采用这种模式的以各类CMS系统为主。例如Drupal和Concrete系统，其基本系统是开源且免费的，但其上的许多插件、主题、模版等是收费的。有兴趣的读者可访问 http://www.concrete5.org网站，其中还有“Marketplace（市场）”频道。

■ 混合模式，既贩卖工具等软件的许可，同时还向用户提供付费服务的模式。比如Ubuntu Linux发行版。

■ 成为平台型项目，并承载自己的互联网业务。这种模式在大型互联网企业中应用广泛。例如Google开发并开源Chrome浏览器，短短几年抢占了微软的很多市场份额，通过在Chrome中默认使用Google搜索引擎而获得极大的收入；再比如Google开源Android，一方面为了遏制苹果iOS的增长势头，另一方面通过预置Google搜索而获得了大量来自移动互联网的流量收入。

显然，有了先驱们的成功案例，作为开源软件参与者，不论是企业还是个人，都可能名利双收。

IT企业为何要参与开源项目？

作为企业，参与或者主导一个开源项目，其最为明显的动力应该是上述的第四个商业模式，即打造一个平台型项目。但就中国的IT企业来讲，我尚未看到有此种实力，或者此种抱负的企业存在，毕竟，打造一个平台需要长期的投入，一般情况需要五年或更长时间。貌似中国没有一个企业有这个耐心来投入五年这么长的时间在一个项目上。

那为什么企业还要参与到开源软件的开发中呢？我认为，谋不了大利就谋点小利，企业主导或参与开源软件，至少有以下几个好处。

■ 提高企业的美誉度。在利用开源项目的同时，也参与到开源项目中，企业的美誉度会得到很大提升。

■ 员工更有激情。因为自己的作品能够公之于众，虽然著作权本质上属于企业，但作为实际的编码者，可以通过开源自己的作品来获得额外的成就感和满足感。这对于稳定开发团队、提高开发人员的积极性会有很大的帮助。

当然，也许过不了几年，中国也能出现实践第四种商业模式的大型IT企业，让我们拭目以待吧！

个人开发者如何利用开源项目获益？

如果你是一名开源软件的开发者，打算利用自己的软件开创一家软件公司，该如何做？第一，我们要确定好自己的商业模式；第二，为自己的开源软件选择恰当的许可证。

如果决定选择双许可证模式，应选择GPL这样较为严格的许可证，它是这种商业模式能够成功的基础。当然，选择双许可证会阻碍产品在商业用户中的推广。尤其是对初生的开源项目来讲，显然是一种两难的境地。MiniGUI之所以可以采用双许可证模式，是因为在成立公司之前和最初的一段时间内，MiniGUI采取的是LGPL许可证，之后在软件足够成熟时才改为GPL许可证。另外，MiniGUI用于功能手机等系统中时，因为这种设备一般使用实时操作系统，缺乏应用LGPL/GPL许可证的技术条件，所以面向这种设备收取许可费也是天经地义的事情。MySQL采用双许可证模式得以成功的原因在于，MySQL AB公司并不会对仅仅用于Web服务器的MySQL商用行为收费，因为这种情况下，商业用户并不会发布MySQL的副本——它只是在服务器上运行而已。

因此，看起来上面提到的第二种、第三种商业模式是最适合个人开发者或者初创公司的商业模式，能够很快地速度推广和迭代软件本身，还能够确保有足够的收入来保证下一步的发展。在这种模式下，应该选择较为宽松的许可证。但大部分开源软件作者，由于并不真正理解开源软件的许可证，所以采取了错误的许可证（指在法律上是错误的）。例如RT-Thread，一方面采用GPL V2许可证，另一方面又承诺不会对商业使用收费。这其实没有解决根本的法律问题，即使用RT-Thread开发的衍生作品，到底要不要遵循GPL？这个问题和是否收费没有直接关系。要解决这个问题很简单，采用类似Apache、BSD或者MIT许可证即可。有读者会问，那为什么不能采用LGPL许可证？就RT-Thread这样的软件来讲，采用LGPL和GPL没有本质的区别，因为RT-Thread的应用场合下一般不支持函数库的动态链接，这导致失去了

适用LGPL许可证的技术条件。

上面提到的最后一种模式，是否适用于个人开发者或者初创公司呢？我的答案是，这种模式是大公司的玩法，小团队或小公司没法做这类事情。

大专院校应该成为开源软件的主力军

一个有趣的现象是，很多开源项目其实是作者在大专院校或者研究机构工作或学习时发起的，比如本文提到的三个国内早期的开源项目。甚至某些开源项目由特定的大学主持和维护，如BSD操作系统、PostgreSQL关系数据库、Minix操作系统等。

从国际视角看，开源软件的发展离不开一些知名大学的参与，BSD和MIT许可证分别由加州大学伯克利分校和麻省理工学院定义，并由两所大学在其众多开源软件中使用，也被其他开源软件广泛应用。值得一提的是，苹果公司Mac操作系统和iOS操作系统，均使用了加州大学伯克利分校开发的BSD操作系统内核。

从现实情况看，在职的程序员，除非因为供职单位支持，否则很难独立发起和维护一个大型的开源项目，但在大专院校和科研机构工作的老师和学生，则有得天独厚的条件（主要是有大量的时间，并可能与科研课题和教学任务相结合）来发起和持续维护一个开源项目。清华大学陈渝副教授主持的SkyEye和ucore两个开源项目就是典型的案例。笔者希望国内有更多的大专院校和科研单位（尤其是教师）能够积极参与到开源项目的发展当中，并成为国内开源项目的主力军。

政府和开源社区应该做什么？

在促进开源项目的发展中，政府要做的就是制定公平、合理的规则，促进相关法律法规的完善。如果知识产权保护力度不够，不仅会阻碍软件产业的发展，也会阻碍开源项目的发展。此外，需要政府支持建立以支持开源项目为己任的非营利性基金会组织。

加大知识产权的保护力度，一方面可以让商业软件在传统贩卖软件使用许可的商业模式下得到良性发展的机会，另一方面可以促使一部分人使用免费的开源软件，进而促进开源软件的发展。

就现阶段而言，如果政府能设立一些奖励基金等奖项，给开源项目的作者以一些奖励，也是不错的支持途径。

开源项目应该以松散、自组织的形式开发和发展，开源社区的存在，为开源软件开发者和使用者提供赖以生存的土壤。开源社区可以是网站、论坛，也可以是松散的交流、展演等。当然，开源社区第一步要解决的问题就是自己的生存问题。

我的建议是，开源社区应该尝试在现有法律框架下，以有限责任公司的形式来做国外开源基金会所做的工作。通过这种方式，可以有效避免无法注册NGO组织的问题，然后从企业（尤其是那些大型互联网企业）当中募集捐款，通过赞助一些开源项目，逐步推进开源软件社区的良性发展。

另外，国内开源社区还需要从使用者社区转向开发者社区，为开发者参与开源软件提供便利，如建立类似GitHub/SourceForge那样的开源软件托管站点，为开源软件项目提供邮件列表、论坛、博客服务等。

结语

将开源项目和商业结合，不管是在自己的项目中使用开源项目，还是靠自己的开源项目来赚钱，都无可厚非。关键是，我们需要尊重开源软件著作权的拥有者，按照开源软件所采纳的许可证办事，只有这样，开源软件才能得到长足发展。合法使用开源软件的前提，就是遵守开源软件的许可证规定的各种义务。

当然，更有积极意义的参与开源项目的方式是，将使用开源软件中遇到的问题或者修正、增强代码提交给开源软件的作者，帮助其改善作品。其实，这是任何使用开源项目的企业和个人都能做到的。

魏永明

北京飞漫软件技术有限公司创始人，他主持的MiniGUI项目，是国内最具代表性的几大自由软件项目之一，目前已成为支持多种嵌入式操作系统的图形中间件产品。

乐趣与高效

淘宝章文嵩谈LVS与商业公司参与开源

记者 / 卢鹄翔

章文嵩是Linux内核开发者，LVS项目的发起人。同时，他也是淘宝开源委员会负责人。在ADC 2012大会上，《程序员》杂志就个人和企业参与开源项目等话题对章博士进行了访谈。

收获开源的乐趣

《程序员》：请你谈谈开发LVS的经过。

章文嵩：LVS是我在1998年5月开始做的，那时我还在读书。在那之前，我写过很多软件，但出于种种原因，那些软件的生命周期并不长，LVS是我目前写过的软件中生命周期最长的一个。

我在1995年底开始接触Linux，并不算早，当时只是装机器给同学们玩儿，并没真正在Linux上写过代码。不过因为使用Linux，便开始接触到互联网上的开源思想，我也读过Richard Stallman的故事，对他的观点很认可。

在我1997年硕士毕业时，曾问老师能否把硕士课题的成果开源出来，但老师说这是国家课题，不能开源。所以，那个软件只能锁在抽屉里，至今也没有发挥出更大作用。

一开始写LVS，我只花了两个星期的时间，完成了一个很简单的版本。它是我的个人项目，与学校无关，所以不需要做任何申请，我就能决定将它开源。

因为我接触互联网较早，所以自己搭建了网站并将源代码放上去。一个星期后，LVS就被一家澳大利亚小公司用起来了。这家公司为社区提供上网服务，那时互联网的带宽比较贵，它通过正向代理服务器缓存之前用户访问的数据，并提供给后来的用户。这样到海外的流量只有一次，对ISP来说可以节约很多成本。但随着社区上网人数增多，一台机器已经忙不过来了，这时它们发现了LVS，然后立刻用起来。公司的负责人写邮件感谢我，说效果很好。我觉得这件事挺有意思，因为自己写的软件被别人用起来了，为他人创造了价值。

在软件的整个开发过程中，逐渐有许多其他人参与。比如一开始，项目的英文介绍过于简单，这时一位美国朋友帮我写了英文的HowTo，后来有其他开发者陆续加入进来，有人贡献过一两个Patch；也有人持续了很长时间，帮助我一起做LVS。

LVS进入官方内核之后，维护工作就相对简单了。以前，每个版本发布我都需要检查与系统其他部分的兼容性。进入官方内核之后这些工作有更多人完成，其上的开发也要照顾到LVS。目前，我们正在实现一些新功能，比如对IPV6的支持是Google的两个工程师帮我完成的。我们也正在把对SCTP协议的处理加进来。这是开源项目的好处，不需要所有事情一个人单干，可以在全世界范围聚集一群有相同兴趣的人一起完成。

《程序员》：LVS是你在读书时完成的，在学校，你是怎样了解到生产环境中真实用户的需求呢？

章文嵩：我认为对于一个开源的项目来说，搜集

《程序员》杂志就个人和企业参与开源项目等话题对章文嵩（图左）进行了访谈

需求很重要，因为真正有用的软件来源于实际应用，不同于闭门造车想象出来的功能。

我确实是见到这样的问题，才有了开发LVS的想法。我在国防科大学读书之余，也帮湖南电信做一些系统集成项目，见到许多服务器忙不过来的情况。这时就需要考虑负载均衡。那时业界也有一些商用方案，常用的是思科LocalDirector。我希望在Linux内核上写点东西，又知道思科LocalDirector的工作原理。所以我当时就花了两个星期写了一个最简单的版本，那是1998年5月。LVS网站上最早的一个Logo，是一个箭头进来分成三叉，跟思科的LocalDirector差不多，只不过我在上面又添了一只企鹅。后来这个Logo变成一群企鹅，是别人给我的启发，因为软件是关于Linux集群的。

在开发LVS的过程中，我收集了很多来自真实用户的需求。即使我没有接触到某些应用场景，但会有其他能接触到这些场景的用户协助我进行测试，这种方式把软件的开发成本降低了很多。

《程序员》：你为什么决定将LVS开源，而不是像思科LocalDirector一样，也做成商业产品？

章文嵩：一套LocalDirector当时价值几万美元，我那时心里有疑问——这东西并不难写，为什么卖那么贵？而我做开源项目纯粹是为了好玩，只要有人用我就很开心，哪怕有时跟用户交流到很晚，需求来了，我半夜还在写程序。

2000年前后，收到了许多用户的感谢邮件。也有公司给我捐送服务器，有个瑞士用户寄了一大包饼干来感谢我。我把饼干拿回家，被丈母娘看到了，她说原来你半夜这么辛苦地工作，就是为了一大包饼干啊。

做这些事是很有成就感的，更多的是乐趣。

《程序员》：对于正在从事或准备从事开源项目的开发者，你有哪些建议和经验希望分享？

章文嵩：我觉得做开源项目，迈出第一步很重要。代码不在于多与少，项目也不在于大与小，先开源了再说。因为开源是指整个的开发过程，并不单单指代码。代码开源只是一个结果，更关键部分还在于整个开发过程。在LVS的开发过程中，虽然没有很多金钱回报，但其他方面回报是有的。比如出国参加演讲一般会有人资助，可以全世界跑一跑。在整个开源的过程中，我自己学到很多东西，比如与人交流，因为这是一个世界舞台，有机会接触到世界上第一流的编程高手。

想想看，当年我只花两个星期写出的代码，会很复杂吗？但就像埋下一颗种子，只要继续投入，就有可能生根发芽，长成大树。

《程序员》：你是怎么想到一开始就把项目文档写成英文？又是如何推广的？

章文嵩：我接触互联网时，很多资料是英文的，自然而然也想到用英文写项目文档。那时候的社区都是通过邮件列表交流，语言也是英文。当我完成了这个项目，就希望请大家看看LVS有没有价值。于是想到发邮件到社区上推广。否则仅是建立一个网站，可能没有人知道，必须要寻找一个渠道，让更多人看一看。

我们从中学就开始学英语，写一些简单的英文不会很困难，只要把想到的东西用英文列出来，大部分老外都能看得明白。当然，现在国内整个技术发展氛围也越来越好，写中文文档我觉得也不是问题。

比如我们的Taocode、盛大和金山卫士的开源平台

背后都有公司支持，肯定会长久做下去。即使在Github上用中文交流也没问题。因为国内受众也很多，用中文交流有时候更直接。当然，如果想发展国际用户，需要写英文文档。

让企业拥抱开源

《程序员》：你在工作中同时负责商业化和开源方式开发的软件，对比这两种方式，你觉得哪种效率更高、质量更好？

章文嵩：我认为开源方式的效率更高、质量更好、成本更低。

在淘宝、阿里我们主要做底层的基础平台开发，其上承载着上千个应用。在完成这个任务的同时，我们将其中一些基础软件项目开源。一方面是为了与同行交流技术，也便于在校学生看到真实系统中的软件是什么样子，对学生来说，还能让他们更多地了解公司。

以TFS来说，中兴通讯的研究员曾经在《程序员》杂志上发表过关于它的技术文章，新浪微博也利用TFS做图片存储。用户在使用的过程中，对软件中存在的Bug进行反馈，这对我们提高软件的可靠性、性能都有作用。

另一方面，开源软件项目对软件开发人员也是一种激励，因为过去公司开发的闭源项目，往往在满足业务需求后，就停滞不前了。

2010年3月，我们确定了第一个开源项目——淘宝分布式缓存跟对象存储系统Tair。当时我们的开发人员表示，需要给他们一段时间，把代码好好整理一下再开源。因为在开源的时候，会标明每位开发者的贡献，每个文件是谁写的，他的名字、邮箱都会标注在代码和文档中。3个月之后，项目正式开源时，事实证明我们的开发人员确实也把代码写得更好。这对整个项目代码的质量是有促进的。此外，外部用户的反馈对软件是巨大的促进，形成了一个良好的循环，因为软件在一家公司面临的需求相对单一。

淘宝内部有一些开源活动，公司给予一定的经费支持，我们也设立一些奖项，比如“开源达人”，颁发奖状、奖牌。这些奖励多是精神层面的。

《程序员》：商业公司是否应该将内部的软件项目开源？你觉得什么样的项目适合开源？

章文嵩：商业公司最终还是要实现赢利，才能在市场竞争中要占据有利地位。选择开源或不开源，与一家公司的商业模式有很大关系，如果一家公司的所有核心竞争力都在某个软件——比如Oracle，它不可能把Oracle数据库开源。另一方面，我们也看到很多真正的互联网公司发起了很多开源项目——比如Facebook，因为其核心竞争力并不在于软件。当然，软件是互联网公司的重要能力之一，但其核心竞争力有可能在其他方面——比如公司的运营模式或数据。

对于阿里来说，最大的价值在于数据。因为数据是日积月累才获得的，别人拿不走。打个比方，建立一个小规模的电子商务网站，技术实现并非十分困难。但完成之后，其上没有任何数据。在淘宝上，每个人有很多交易行为，这些历史、信誉记录会让大家觉得交易更可靠。而在一个全新的平台上，用户交易的可能性会低很多。按照马云的说法，淘宝、阿里未来是一家数据公司，数据才是最有价值的。

《程序员》：淘宝确定项目开源的流程是怎样的，在项目的开源过程中有哪些值得注意的地方？

章文嵩：我们成立了淘宝开源委员会，通过这个委员会管理项目的开源流程。将公司的一个产品开源，首先要得到直接主管的许可，提出书面申请后还需要公司另外几个技术主管的书面认可。开源委员会会对许可证的选择、代码测试的程度、文档的程度给出建议并进行备案。淘宝的开源项目中，使用GPL许可的比较多，当然有些开发者比较喜欢用Apache许可，我们也不反对。

在对外开放的过程中，我们内部的技术部门还建立了审核机制。安全团队负责审核代码开源后是否会被他人滥用，对我们现有系统发起攻击。比如一些密钥，或者加密的方法，就不适合放在开源的系统中。在我们的实践过程中，证明这样的流程是可以保证现有系统安全的。

OpenResty发展之路

OpenResty项目创建者章亦春专访

记者 / 杨爽

章亦春（网名agentzh）是一位极度热衷于开源事业的程序员，是开源项目OpenResty的创建者。曾供职于中国雅虎和淘宝网，从事过高性能Web系统、前端AJAX应用框架、系统运维工具、自动化测试工具和网页智能抽取等方面的开发工作。目前在美国CloudFlare公司从事系统开发工作，和他的妻子一起幸福地生活在加利福尼亚。

站在巨人的肩上

《程序员》：请你谈一下创建OpenResty项目的背景。

章亦春：OpenResty是在2007年底作为中国雅虎搜索部门的项目进行开发的。当时OpenAPI很火，部门领导希望建立一个通用的高阶Web Service平台供第三方网站的站长快速构建Web应用。只是，不久之后，OpenResty的开发重心转向了支持中国雅虎内部的产品和业务，特别是“全能搜索”。最早的OpenResty是用Perl 5、Haskell以及一些C代码来实现的。

2009年，我从中国雅虎转入淘宝网时，我和同事王晓哲决定将这个项目推倒重来，基于Nginx和Lua来开发新一代的OpenResty。新版OpenResty把性能放到首位，特别是I/O密集型Web应用在高并发访问下的性能。新版本又被称为ngx_openresty，以显示它与Nginx之间的密切关系。新的实现几乎全部使用C语言来编写。之后的两年中，OpenResty的开发主要是为了满足淘宝数据部门的量子统计产品的业务需求。

由于这个项目是顺应公司的业务需要而产生的，所以无论是项目本身还是开源过程都得到了当时公司领导的支持。现在回想起来，这点确实是比较难得的。

《程序员》：可否分享一下设计和开发OpenResty过程中的心得与经验？

章亦春：OpenResty的设计和开发自始至终都是由现实的业务需求驱动的。中国雅虎和淘宝网的用户流量要求我们必须在性能上追求极致，同时在易用性上达到比较理想的平衡。时至今日，OpenResty中开发任务的优先级安排仍然是由用户的实际需求决定的。用户的需求可以确保OpenResty一直走在正确的轨道上，即使在必要时推倒重来也在所不惜。

OpenResty直接以Nginx和LuaJIT这样优秀的软件作为基础，在开发过程中不断地将问题甚至补丁反馈给项目。可以说，我们是站在巨人肩膀上的幸运儿。

测试驱动是我一直坚持的开发方式。我们在测试工具链上投入巨大，开发了Test::Nginx、etcproxy、mockeagain等一系列有趣的测试模块或测试工具，同时也引入了valgrind等现成的优秀工具。各个组件都有比较完整的测试集，它们是项目的生命线。完整的测试集使得我们能放心大胆地对软件进行各种修改，包括较大的重构。

当然，随着各组件测试集规模的日益膨胀，如今在一台PC或者笔记本电脑上运行所有的测试已不再可能，因此我设计了一套类似MapReduce的集群调度脚本，使测试任务可以动态分配到Amazon EC2云服务上并行运行。感兴趣的读者可以到qa.openresty.org网站上看EC2测试报告，相关的调度脚本可以在opsboy项目中找到。

《程序员》：这个项目各阶段的里程碑是什么？有哪些成功的应用案例？

章亦春：OpenResty软件包本身并没有像样的里

程碑列表，只是其中的各个组件都有远期和近期的TODO列表。

现在，国内外许多家公司已将OpenResty用于生产服务。例如，网易财经和腾讯财经的一些流量很大的Web Service接口中都有OpenResty比较成功的应用；淘宝的量子统计以及淘宝直通车报表项目也大量使用了OpenResty；去哪儿网的许多数据接口以及Web请求的安全过滤也使用了OpenResty；CloudFlare的Web应用防火墙也开始越来越多地使用OpenResty的核心组件ngx_lua。

兴趣是源动力

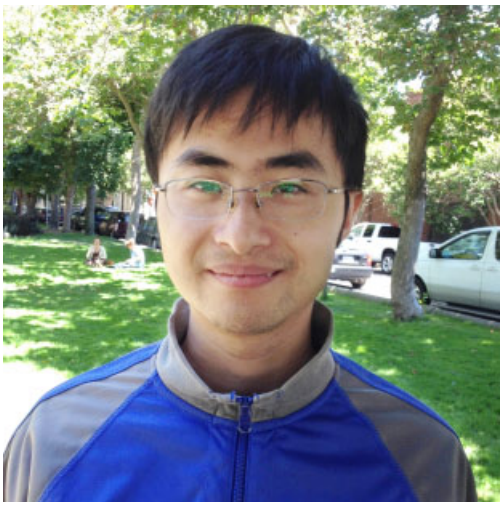
《程序员》：OpenResty项目核心团队的组织结构是什么样的？是如何吸引国际开发者的？国外技术人员主要参与了哪些工作？

章亦春：OpenResty团队的组织方式是相对松散的，因为它本身就是一个形式上比较松散的软件包，包含了标准的Nginx核心、一系列Nginx模块及一系列Lua库。这个软件包是由我负责打包和维护的，其中ngx_lua模块等核心组件由我和王晓哲共同维护，项目贡献者为数众多。其中收录的标准Nginx核心由Nginx官方团队维护，我们经常向Nginx官方团队报告各种Bug、提交了许多补丁；而其中的大多数补丁都已为Nginx官方所接受。其他组件多数是由我和王晓哲维护的，另一些是由我徒弟支家乐在我的指导下编写的，还有一些是由Piotr Sikora编写和维护的，同时还收录了Maxim Dounin和Sergey A. Osokin等其他开发者的Nginx模块。

由于我们坚持在代码和文档中使用英语，所以吸引国外开发者是很自然的事情。同时，多年来我常在国际开源社区中厮混，和国外的黑客交流起来比较自如。开源世界有它自己的文化和规则，只有熟悉这些东西，才能真正融入其中。

我不会按国籍区分贡献者，国内外的开发者都做出了了不起的贡献。其中有很多有意思的故事。

有位来自东欧国家的黑客朋友，帮助OpenResty做了许多很棒的开发工作。实际上，他在当地经营着一家自己的公司，但他深深为开源方面的工



章亦春说：我及时完整地回答用户邮件成为了推广项目的好方法

作吸引，甚至达到了废寝忘食的境地，最终导致自己的公司出现亏损。在经历了好几个月的亏损之后，他很郑重地把他所有的Nginx开源模块都托付给我维护，表示决定暂时退出开源界，集中力量经营公司。可没过多久，他又忍不住提前归来了。至今，他在Nginx社区中仍然非常活跃。

还有一位重要的贡献者是来自英国的黑客。他当时在土耳其首都当英语教师，由于当地很缺英语教师，所以工资很高，这使他可以有大量的时间从事自己喜欢的软件开发工作（包括开源方面的开发），而不用花费太多精力在生计上。虽然他从未在任何一家IT公司上过班，但他在技术上很有自己的梦想和追求。我时常想，那些上过他英语课的土耳其少年，或许很难想象他们眼前的这位其貌不扬的老师，在技术领域是如何影响到世界上那么多家互联网公司的。

开源团队人来人往很常见，也没有什么时间点，与公司团队有很大不同。大家都是按照兴趣或根据自己公司的实际需求来工作，我所要做的便是努力让大家开心，及时审核和应用大家提交的补丁。遇到意见不一的情况，需要耐心沟通，以诚相待，努力寻找大家都能接受的解决方案。

予人玫瑰，手有余香

《程序员》：你是怎样推广这个项目的？有哪些技巧吗？

章亦春：我从没想去专门推广这个项目，只是希望自己做的工作能对他人有用。虽然我也曾经在淘宝D2前端论坛、北京OpenParty聚会、PerlChina北京聚会等场合介绍过这个项目，但这些分享所产生的推广效果是比较有限的。真正起作用的是我及时而完整地回复每一位用户的邮件，避免用户的工作出现阻塞，久而久之，这自然会在用户间口口相传。当然，随之而来的是越来越多的用户邮件和我不断增加的回信工作量。

《程序员》：OpenResty在何时得到了除中国雅虎和淘宝外的商业公司的关注？这些公司有没有对OpenResty项目提供相应的支持？

章亦春：OpenResty诞生于商业公司内部。在它足够成熟之后，其他商业公司也自然开始关注。支持主要有几个方面：一是提供人力直接参与开发和测试；二是直接提供工作机会让贡献者可以继续工作在这个开源项目上；三是提供少量的资金捐赠或者赞助开发。

加入CloudFlare，全职做OpenResty

《程序员》：为什么选择加入CloudFlare公司？能否讲述一下其中的过程？

章亦春：CloudFlare是OpenResty的应用大户。在过去的几年中，CloudFlare的系统工程师Matthieu Tourne经常向我提交补丁，与我也有很多技术上的交流。因此，当CloudFlare向我发出工作邀请时，我并没有感到特别意外。

最初CloudFlare公司邀请我加入时，我已从淘宝辞职，和我妻子在福州过着所谓的“田园生活”。当时，我没有正式的工作，在家继续推进OpenResty等开源项目。这是我非常期待的一种生活和工作状态，事实上，在北京工作的头两年就有了这种打算，因此当时我并不急于找工作，于是谢绝了这个工作邀请。但后来CloudFlare的几位创始人在得知我的“田园计划”后，提出让我在他们的办公室里继续做我一直在做的事情，只是“田园生活”的地点从福州搬到了旧金山。

去年他们还邀请我们去旧金山玩了两周，并且参观了他们的办公室。这次访问让我们确信，这是一个更适合“田园生活”的地方，于是决定接受这份工作。

《程序员》：OpenResty项目现在的维护情况怎样？未来发展方向是什么？

章亦春：如今我在CloudFlare仍然是全职工作在OpenResty项目上。其核心组件ngx_lua模块的开发非常活跃，除了我之外，一淘量子团队的王晓哲和淘宝核心系统部的静龙仍然在积极地开发重要的新功能。

未来的发展方向主要有两个：一是能够非常灵活地定制和扩展Nginx服务器的功能，自如地通过Lua语言操纵Nginx核心；二是更加完美地支持构建高性能的Web Service乃至构造对性能要求极高的完整Web应用。

关于国内开源，我想到的……

《程序员》：你怎样看待国内的开源现状？对国内从事开源软件开发的后来者，有哪些建议？

章亦春：国内的开源事业正处于蓬勃发展的阶段，近几年涌现出大量的开源项目，有的源自商业公司，有的则是工程师在业余时间产出的作品。遗憾的是，许多开源项目止步于开放源代码，却不知开放源代码只是万里长征的第一步，后续的文档建设、与用户的积极互动等都需要巨大的时间和精力的投入。开源精神的核心在于“分享”与“协作”，我们似乎比较容易做到“分享”，特别是“分享”源代码本身；而在“协作”方面，无论是国内开发者之间的“协作”，还是与国外开发者之间的“协作”，可能都还有比较长的路要走。

对于国内刚开始涉足开源的朋友，我还有一个建议就是一定要努力学好英语，以便能够自如地与世界各地的程序员交流和协作。

开源是种生活方式

文 / 方越

走上开源道路

清晨六点钟，被儿子闹醒，第一件事是打开邮箱，迎面而来的是数以百计的邮件，我知道新一天的工作开始了。

这是一个典型的场景，过去四年多的时间，我一直在家工作，全职参与Apache旗下多个开源项目的开发。以开源社区为核心，以maillinglist、IRC、JIRA、SVN、Git、Maven等为工具，参与全球化协作的开源项目。

走上Apache开源道路，要从2006年谈起，当时源于ObjectWeb社区的Celtix和源于Codehaus社区的Xfire合并，成为了Apache旗下的一个新项目CXF。我当时所服务的公司IONA是开源项目Celtix的主要推动者，也是Celtix的代码提交者（Committer），因此成为了Apache CXF的Initial Committer。从那之后，我一直从事开源软件开发工作，期间伴随着IONA被Progress收购，以及Progress成立独立的专门从事开源项目的子公司FuseSource。根据项目发展的需要，我从事的工作也从最初的Apache CXF扩展到了Apache旗下的多个开源项目。目前，我是Apache CXF、Servicemix、Karaf、Camel、Felix的Committer（在Apache开源社区中我使用Freeman Fang这个名字），并且是Apache CXF、Servicemix、Karaf的PMC member（项目管理委员会成员）。

我想读者最为关心的问题是，全职从事开源软件的开发，靠什么来生活？是完全凭兴趣吗？如何保证自身的可持续性发展？

Ok，要回答这些问题，必须先谈一种基于开源的商业模式——围绕开源社区和开源软件，通过雇用主要的开源项目开发者，向外提供付费的商业服务，这种付费服务包括订阅、咨询、培训等。我的雇主就是提供类似服务的商业公司。这种开源商业模式在基础软件架构领域（如开源操作系统、开源数据库和开源基础中间件）已被证明是一种行之有效的方式，在欧美市场被广泛接受。但在国内，这种商业模式还没有被普遍认可，很多人还认为开源就是免费，不愿意承认特定领域专家的价值，这种现象很值得深思。

在我看来，这种开源商业模式下，开源项目和开源社区因为有了商业公司的介入更加活跃，开源开发者有了稳定的收入获得了可持续发展，付费用户由于有了商业支持能更有效地使用开源软件，非付费用户也能从活跃的开源社区获得支持，构成了一个共赢的生态环境。

回到本文开头，我的日常工作是：阅读大量我所从事的Apache开源项目的邮件和公司论坛帖子，回答社区用户问题；实现新功能或者修复Bug，提交代码到Apache代码库中。同时，如果付费用户提交了问题，那么高优先级处理付费用户问题，但实现的新功能或者修复的Bug都会第一时间回馈到相关Apache项目的代码库中。

为什么从事开源项目

如果一个人能很长时间做一件事情并乐此不疲，那么只有一个解释，他很喜欢做这件事。我喜欢开源的原因很多，且听我一一道来。

原因之一：开源意味着最大程度的分享，降低了知识学习的壁垒，给了后发者赶超的机会。

在从事开源软件开发的过程中，我经历了这样一

个心理变化，从很自信到很不自信到慢慢找回自信。由于有机会和世界上最好的一群软件工程师一起工作，我可以很清楚地看到自身的差距，但开源意味着一切都是透明的，不仅仅是代码，公开的社区讨论也能给你很大的启发。时间长了，你会看到自己的进步，慢慢找回自信的过程就是进步的过程。对于一个喜欢技术、追求技术的人，开源是一个能提供无限可能的大平台。

原因之二：开源给了我最大程度的自由。

我不用去办公室工作，能自由安排自己的工作时间。在北京这个大都市里，不用去办公室意味着每天能节省2~3个小时的通勤时间，我可以用这个时间来学习、健身、娱乐，甚至加班。有了孩子之后，我每天早7点之前开始工作，通常下午4点我会将工作停下，接管照顾孩子的任务。在晚上8点孩子睡了之后，我会最后再工作一会儿。

随着Internet等基础设施的飞速发展，伴随着大量辅助工具软件的出现，远程办公成为可能，越来越多的工作由集中式变成分散式，我们的工作模式正在改变。这种模式降低了大城市的交通压力，办公场地的压力，更加节能环保，同时也提高了工作者的生活质量。

原因之三：开源给了我成就感。

每次获得一个Apache项目的Committer权限，都是一次被认可的过程。通常意味着你要不断地提交高质量的Patch给该项目，积极参与该项目的社区讨论，之后该项目PMC才会提名和投票表决。每当收到一个项目的Committer邀请，都会带给我巨大的喜悦和强烈的成就感。此外，参与社区讨论，回答别人的问题，也是一件非常愉快的事情，这真的是可以上瘾的。

从事开源，就有机会和最好的工程师一起工作，做领先的技术和项目，这些项目不仅能直接服务最终用户，还能成为其他项目的重要组成部分。例如，Apache Geronimo选择基于Apache Karaf来实现OSGi容器；还例如Apache CXF作为但不限于JAX-WS协议的一个实现，得到了很广泛的应用。多个JEE服务器如Apache Geronimo、JBoss、JOnAS和Pramati选择CXF作为自己的JAX-WS实现（从JEE5开始则要通过TCK认证且需要有JAX-WS的支持）。

原因之四：开源让我更好地了解世界。

从事开源项目的开发人员，来自世界各地，具备不同的文化背景、教育背景和思维方式。通过和他们交流，我更好地了解了他们的技术水平以及引发了我思考人家为什么会达到这样的技术水平。

举个例子，西方的工程师从事开源软件只有一个理由——兴趣。英文里有一个单词是Geek，翻译成中文是极客，代指狂热于技术、思维异于常人的人群。用Geek来形容开源开发者再合适不过了，他们对技术的追求无止境，他们喜欢创造新的东西，他们渴望改变和引导技术发展的进程。对比国内研发工程师经常忧虑一些现实的困境，他们从来不考虑30或者35岁之后老了没有竞争力的问题，他们很少考虑转型做管理者。他们通常年龄在35岁或者更大一些，10余年的行业技术积累让他们知识的深度和广度都很好。由于社会文化真正认可技术的价值，他们只凭做工程师就能生活得非常好（真的是非常好），他们要考虑的内容相对单纯一些，就是从兴趣出发，完全以技术为导向。当一个人真正对自己从事的工作有极大兴趣、引以为豪且有多年的经验积累之后，其创造力和生产力是惊人的。

技术专家的产生需要土壤，需要社会价值观作为导向，让有天赋的研发人员愿意长期从事技术积累且无后顾之忧，并能获取相应的回报是产生群体性技术专家的基本前提。如果我们的技术人员都在考虑找机会转型，那么恐怕永远只能做低水平的简单重复，离诞生伟大的软件公司这个目标会越来越远，与和世界一流的软件水平之间的差距会越来越大。

原因之五：开源已经、正在和将继续改变软件业的发展以及我们的生活。

基于成本优势；代码可见，可深度定制；系统高度自由，无需绑定到特定厂商；更敏捷等原因，越来越多的公司采用开源软件和开源架构，这种趋势也使更多的商业公司开源自己的产品，以适应市场的发展。

且不说国外开源应用风起云涌，国内也有很多公司重视开源技术，采用开源架构，例如国内某著

名电商互联网公司。该公司重视技术积累，有自己的给力技术团队，有能力对开源解决方案深度定制，快速满足自身需求。该公司的系统每天都在承受超大规模的并发请求但依然很稳定。我一直认为在大压力复杂应用环境中，只有自己是自己的上帝，开源给了你主宰自己命运的机会。这个例子说明真正重视技术积累，有给力技术团队对企业自身的应用是多么的重要。

为什么国人参与开源力度不够大

参与Apache开源项目以来，有一个现象非常困扰我，就是很少有同胞在相关项目邮件列表/论坛问问题，是国人很少使用这些开源项目吗？显然不是，以Apache CXF为例子，如果我们使用Google Trend来查看关键字CXF在Google被搜索的统计，我们可以看到最多的搜索请求来自中国，搜索者使用的语言排在第一位的是中文；同时，我可以在互联网上搜索到大量的关于使用Apache CXF的中文技术博客。以上都能证明，Apache CXF在国内被大量使用，但我为什么在Apache CXF的邮件列表中很少看到中国人问问题，更不要说参与讨论、解答问题甚至打Patch了。我在这里想尝试分析一下为何同胞很少参与讨论，并帮您打消顾虑。

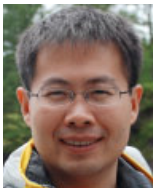
我认为最主要的原因是文化上的差异，它首先表现为语言上面的障碍。开源社区的参与人员来自世界各地，为了保证大家的交流，采用的语言是英语，毕竟英语目前是国际上最为通用的语言。在线下交流中，我了解到，很多人因为要用英语问问题而迟疑，甚至放弃问问题。其实大可不必，因为Apache社区很多参与讨论的人员的母语都不是英语，这并不妨碍大家用英语讨论技术问题；再者，我和欧美同行交流过语言方面的问题，他们（尤其是英语是母语的同行）认为很多中国人过于低估自己的英语交流水平。实际把自己的问题用英语描述清楚并没有那么难，别担心，没有人会像针对我们的英语考试中短文改错那样对待你的邮件/帖子。毕竟，我们掌握英语的程度远远超过西方人掌握中文的程度。另外，多参与讨论，对实用英语的挺高也很有帮助。

文化上第二个差异体现在不同的学习方式上，我们擅长的学习方式是有系统的教材，有标准的大纲，最好再有大量的练习题来强化知识点。我们习惯于这种学习方式，我们对规划好的知识点可以掌握得非常好，具体的体现是我们很擅长考试，但我们很少主动问问题。大多数人在考完试之后，马上就会忘掉之前死记硬背的东西。西方人要学习一个东西，习惯自己找材料，找问题，讨论问题，因此，讨论问题本身就是他们所习惯的学习方式的一部分。而通过这种方式获得的知识更让人记忆深刻。我认为他们更善于学习新东西，尤其是需要自己主动学习时，他们的学习方式让他们更有优势。我们害怕没有面子，害怕问出不靠谱的问题，回避交流。不要这样，要更开放一些，善于利用Apache社区这个很好的资源，和技术专家讨论问题。况且，英语有一句俗语，“No question is stupid”，就是在鼓励大家问问题。

本文简单总结了我是如何走上Apache开源道路的及从事开源项目的一些感想，很多思考来自于新浪微博和各位朋友的交流过程（我的微博@Freeman小屋）。如果本文能鼓励更多的同胞积极参与开源项目，写作本文的目的就达到了。

另外，本文一家之言，纯属个人观点，不代表任何机构、组织和商业公司。

最后想说，能有机会全职参与Apache开源软件的开发，是我的机会好。我不是技术专家，我和技术专家一起工作，我一直在努力，我是Freeman。



方越

自2005年来全职参与Apache CXF、Apache Servicemix、Apache Karaf等多个开源项目，擅长Web Service、SOA、ESB、OSGi等领域。

从BlenderCN，看开源社区建设

文 / 罗聪翼

社区除了泛指我们的居住地区，还特指一个拥有共同文化而居住在同一个区域的群体，前者强调的是地域，而后者则强调文化。不过无论侧重点偏向哪边，社区都赋予群体内部成员之间的文化维系力和归属感。

开源社区是依附于开源软件同步建立的服务性社区，旨在为用户和开发者提供一个在线沟通交流的空间，因此开源社区在推动开源软件发展的过程中起着巨大的作用。在国外，社区通常由开发者直接参与建设，以论坛、Wiki、邮件列表和IRC聊天室等形式组成，使开发者在与用户的沟通交流中获取新的需求，同时也在答疑解惑中实现项目的用户支持。但在国内，社区一般都是由软件的使用爱好者组成的，对于非本地开发的项目，以本地化支持如汉化软件和Wiki为主。社区的维护主要靠个人，因此生存力很弱，并且很难聚拢开发爱好者。

对于大部分的国内开源项目，开发者更多专注于自己在项目上的研发，而缺乏对社区的建设意识。是否能为自己的项目开拓市场、发展用户，甚至产生有效的商业模式去支持自身发展，都是在听天由命。当然这也是由于大部分国内的开源作者，多为利用业余时间去验证某个想法的独立开发者，所以在人力有限的情况下只能在自己的项目托管主页上提供简单的用户交流窗口。这种做法很容易拖累项目导致软件本身的发展滞后。

Blender是一款开源的3D软件，其主要维护机构为荷兰Blender基金会。一群国内爱好者于5年前建立了中国BlenderCN爱好者开源社区，社区以提供本地化汉化和应用交流为主。那时，在社区里注册的Blender用户10个中有9个都是来图新鲜玩玩的，而剩下1个能坚持下来的用户，也不一定能参与软件的开发或者社区建设。就在这样有限的爱好者中，社区聚集了5个忠实用户，在获得了官方的社区建设支持许可之后，决心将大家热爱的这款优秀软件坚持推广下去。

在这几年的发展中，社区经历了多次变革，经历了几次起死回生。2009年，由于服务器赞助者撤走了免费资助，社区一度处于网站被迫关闭的状态，大家只能商量自发凑钱，在勉强维持了几个月之后才寻找到了新的空间赞助。2010年，由于备案问题，网站被限制了国内访问，在数据迁移的过程中，因技术疏忽，几年内积累的论坛数据损坏，社区的历史被抹平回到了起点。好在社区的成员群都还在，大家很快重新架设了新的论坛和数据库，几周内注册人数就达到了400人，社区很快得到了恢复。2011年，社区的博客团队因故退出，一直倍受欢迎的中文博客也被迫关闭。因为人手不够，所以博客更新就只能处于无限推迟状态。2012年，存放Wiki数据的海外服务器供应商恶意违约，在未通知的情况下擅自迁移主机，导致Wiki数据彻底丢失，虽然尝试通过备份和网页快照的方式找回了部分页面，但大量的宝贵资料依然无法挽救。

即便如此，经过了近5年的发展，社区依然留存了大量的忠实爱好者，目前已经拥有了几千名注册用户，稳定的在线交流人群有上百人。这对于一个相当小众的开源软件，这个用户数字已属不易。同时，社区也在从纯用户社区逐渐转型成为一个本土化开发型社区，用户中已涌现出了不少

本土开发者，他们不仅贡献了软件和Wiki的完全汉化，还翻译了大量教程，参与了Bug测试和跟踪、GsoC开发项目、功能开发等。在这样反复的折腾与磨难中，社区建设者总结出了不少社区维护的经验。

尊重开源

尊重开源是社区的主旨，也是社区中推行的文化，一旦形成良好的文化，就可以让每一位参与者得到熏陶，使参与更自然。开源精神不是拿来主义据为己有，而是秉着自由、互助和尊重的观念，在社区中坦承交流、互相学习。社区之于用户的目的就是提供一个高于互联网的学习空间，一个交换学习资源的活动场所。而对开发者来说，社区是用来获取用户反馈、以最低成本了解用户真实需求的地方。用户尊重开发者的劳动，为其提供建议和Bug反馈，开发者尊重用户的需求，在不断地交流和沟通中，完善并强化自己的项目。因此作为社区建设者，最重要的工作就是要减少两者的沟通成本，降低交流落差。社区做的最多的就是，尝试提供本地化的学习资源（无论是汉化还是原创），浅显易懂的用户使用手册和实例教程是留存用户最简单的方式。Wiki和论坛在这里扮演了极为重要的角色，公开编辑权限也是一种基本的开源方式。论坛鼓励大家自由发言，管理员只负责过滤和删除广告等与社区话题无关的内容。

不拘束于本地

除了做好本土化维护，还应与其他地区的兄弟社区产生互动和交流。Blender的活跃社区主要分布在欧洲、美国和东南亚，因此管理团队首先尝试去与台湾、澳门和新加坡等拥有共同语种的社区分享资源、交换链接和邮件列表。对于欧美等英文社区，则使用作品这种世界语言的方式，参与沟通和讨论。Blender的用户群正在逐渐形成一种社交型社区，这种沟通基于各大SNS平台，如Facebook、Google+和LinkedIn等。因此，用户会逐渐发现，要想融入一个社区其实很简单，找到社区中的国际语言，参与真的很容易!

稳定的管理团队

对于社区的建设和维护，应尽力使用公开透明的方式进行，虽然管理团队的成员拥有不同的背景，但大家都尊重开源精神、彼此信任、各司其职，重在自愿参与，不强制要求每个成员应该怎么做，而是鼓励对方如何去尝试。当一个社区从几十人发展到几百上千人时，管理团队就更需要拥有责任心和恒心，此时就应该拥有一名总决策人，他需要具备全面的技术知识，建设社区的激情，善于聚拢的社交能力，但他的权利也不能是绝对的，因为在一个非盈利性开源社区中，管理者和参与者的界限应该尽量模糊，平等是一个基本条件。不过针对关键问题时，决策人要有明确而鲜明的立场去稳定团队。例如，BlenderCN社区曾经也考虑过商业化，但得益于以创始人为首的管理团队，坚持以非盈利的方式去维护社区，即使有商业的参与，也要谨慎地选择并在社区的外围接受合作，这一点对于稳固一个开源社区的核心文化价值十分重要。因为BlenderCN定位于一个非营利性组织，所以就不应该利用社区本身去创造价值，但可以基于社区，帮助更多的人去实现价值。

高质量的线上线下交流

本地化社区的优势是，可以为成员提供线上和线下交流的机会。BlenderCN社区分别在上海、北京和成都举办过多次大型的线下交流。线下交流可以小型聚会方式举行，借助大型活动协助推广的方式，针对不同的人群开展对应的话题交流。在上海，社区和上海建桥学院合作开展了校园开源知识讲座，面向以在校学生和老师提供教育类的资源进行分享。与盛大创新院合作交流时，面向的是企业级开发者和商业用户，因此提供了新技术分享以及解决方案分析；在北京，与中国传媒大学合作交流新课程设计，并建立了学生实践试点合作；在成都，结合HTML5技术大会，分享了如何将Blender结合到行业应用当中。线下交流可以考虑有规律地进行，结合现有的活动和宣传渠道，在推广社区和产品的同时，也能自然而然地吸引更多的用户加入进来。

将社区由用户转向开发

当用户群体逐渐壮大时，有限而且跨国的开发者将无法更快地满足本地用户的需求。此时，社区便推动一部分技术爱好者转型，例如直接参与代码开发、功能改进、提交Bug并跟踪测试，参与新版本的功能性测试、文档编写，甚至通过付费捐助的方式间接地支持开发。除了鼓励社区中的现有成员参与，管理团队也利用每年由Google举办的Google Summer of Code项目机会，在国内高校中寻找学生开发者，并借此推广社区和软件本身。2011年，北京大学的一名学生便成功地申请到了GSoC开发项目，在3个月的开发周期内顺利完成了全部功能，他不仅获得了十分丰富的项目实战开发经验，还拿到了一笔数额不小的奖学金。

帮助有能力的社区成员寻找商业模式

在经历了初期的建设支撑住了生存，度过了中期的发展完成了转型，此时社区已成为一个不依赖管理团队用户大量参与的活跃社区，并且能自行扩展和吸纳新的成员，这也就是一个逐渐成熟的社区。而通过线上和线下交流，成员之间也能形成一定的关系网。此时社区的主要任务可以从吸引用户逐渐转变为吸引成熟的资助者，从人脉和资源上，帮助有能力的成员寻找适合自己的商业价值。

从2009年开始，社区中就已有一部分成员聚在一起，开始尝试组建工作室并接单，社区在背后提供技术支持和人力资源推荐。无论是自由职业者还是工作室团队，这种社区型的威客商业模式成功帮助了大量的社区成员。当成员在分享自己的收入和成绩时，更多的人也会参与进来，加入学习和交流。与此同时，社区的培训和能力认证体系也应运而生，2010年，社区成员出版了中文教程书籍及视频教程，线下社区则组建了技术交流沙龙。学习者获得了技术的提升，能力的认证也建立在线上的口碑，论坛上的作品成为了每一个人的技术简历。给予者赢得了社区的尊重，培训的收入一部分被用于捐助社区网站的持续维护，而受惠的社区成员也知恩于社区的帮助，更愿意去回馈，这就能形成一个良性循环的发展模式。

小结

社区建设其实也是一种开源项目，目的不仅仅是建立一个具有社会意义的组织，更要塑造出一种开源社区文化。这种文化具有传播力和凝聚力，其社会影响力不亚于一般的企业。用户型社区的使命是带动并推广开源项目的使用，而开发型社区的目的就是聚拢更多的价值劳动力，帮助创作者完善项目。如果要实现这些目的的话，就需要善于平衡两者的发展，并且要在适当的时候选择转型，这也是开源社区的基本定位。如果定位不准的话，组织者对定位方向一旦动摇，社区也可能会摇摇欲坠。当然这里也不是要建议去回避独特的定位，只是希望能分享一些经验，为社区建设者提供更多的思路。

开源本身就是一种加速创新的途径，新的技术都可以尝试在开源项目上开辟实验田，这对科技的帮助是极大的。但有时开发者和用户会有两种不同的需求方向，因此如果能在加速技术创新的同时，让开源社区融合进来，技术转换为产品的周期将会有可能大幅缩短，因为适当的交流和反馈可以帮助开发者少走不少弯路。如此看来，这是一个互相推动的过程，这种互推可以存在于技术和创新间，也可以应用于企业和个人间。

未来是一个属于开放态度的时代，软件本身的许可已不能制约商业价值的实现，只有更积极地去跟进开源、拓展社区、发展用户甚至做了少许的贡献之后，你也能发现其背后所蕴含的巨大价值！

罗聪翼

BlenderCN开源社区管理团队成员之一，现任职于成都优聚软件（GoodTeam Studio）负责开源化项目开发流程。

你必须了解的开源法律知识

文 / 董颖

当开源软件/代码摆在我的面前

开源代码对程序员来说有极大的吸引力，而在如此巨大的诱惑力之下，可以随意使用这些开源软件和代码吗?

当然不是，开源软件附随许可协议（License），使用开源软件或代码须符合其许可协议的要求。这是因为软件和代码本身受其作者所享有的著作权保护。著作权是一种知识产权，它在开发者完成软件开发、代码编写时就由其作者自动享有（并不需要经过注册或其他程序）。对享有著作权保护的软件和代码，他人不能随意使用；对软件或代码进行复制、修改、分发或传播，必须得到著作权权利人明示的许可，否则将构成著作权侵权行为。

这里需要澄清以下几个概念。

■ 自由和开源软件（Free and Open Source Software，FOSS）≠没有任何条件或无须承担任何法律义务可随意使用的软件。“自由软件”中的“自由”二字容易引人误解。其实所谓“自由”是指软件著作权权利人授予使用者四项“自由”权利——使用者对软件可以进行使用（自由之零）；对源代码可以修改（自由之一）；以及对该软件及其修改版本可以进行二次分发（自由之二及之三）。此外，软件著作权权利人仍可以对使用者提出其他要求，需满足其他什么条件、承担其他什么义务；只要不与上述四项主要的“自由”权利相冲突，使用者仍须遵守；这些条件和法律义务一般都规定在软件附随的许可协议中。

■ 开源软件≠自由软件。这是因为获得源代码是享受上述“自由之一”（修改源代码）的前提，因此“自由软件”一定是“开源软件”；但反过来“开源软件”并不一定是“自由软件”，因为它并不一定对使用者提供上述所有四项“自由”权利。

■ 免费软件（Freeware）≠自由软件（Free Software）。虽然在英文中是同一个词“Free”，但它对应了“免费”和“自由”两个意思。“自由软件”和收费与否并不相关，这是因为上述四项“自由”权利与收费与否并不相关。因而“自由软件”不一定是免费的软件，也可能是收费的商业软件。

应当明确：不论其收费与否，使用开源软件均须满足其许可协议的要求。

用开源软件 代码会产生法律纠纷吗

任何一个开源项目都需要维护其开源“生态环境（系统）”。虽然有很多商业公司采用开源软件或代码进行产品开发，但不尊重“生态环境（系统）”的行为无疑会受到开源社区的排斥。现在有越来越多的开源法律保护组织在采用法律手段保护开源“生态环境（系统）”。著名的开源法律保护组织如软件自由法律中心（SFLC）、自由软件基金会（FSF）、gpl-violations等。

已发生的法律诉讼已有多起，例如SFLC在美国针对使用Busybox的消费类电子厂商发起的一系列诉讼。Busybox是一款采用GPL协议的嵌入式开源软件，因其专为嵌入式系统设计、所占资源少但功能强大，而被誉为“嵌入式Linux中的瑞士军刀”。有很多消费类电子厂商的产品里使用了Busybox，但并未遵守GPL协议将其代码开源，

于是SFLC将诉讼的矛头对准了这些厂商。这包括2007年起诉Monsoon Multimedia、Xterasys、High-Gain Antennas、Verizon Communications，2008年起诉Bell Microproducts、Super Micro Computer、Extreme Networks，2009年起诉Best Buy、Samsung、Westinghouse等14家公司。

当然，采用开源软件还可能陷入其他法律纠纷。例如，Google及采用Android平台的下游厂商（统称Android阵营）所遭遇的专利诉讼：其中包括苹果针对Android手机厂商（HTC、Samsung）的专利诉讼，微软针对Android阵营的专利诉讼（HTC、Amazon、Motorola、Barnes & Noble、代工厂商富士康和英业达）。虽然Android是开源软件，采用Android系统无疑降低了开发成本，但Android阵营所遭遇的专利诉讼，也显示了开源系统面临的隐性成本——法律诉讼成本。

在众多法律问题中，本文重点关注开源合规性问题，即使用开源软件/代码是否符合许可协议的规定。以上开源软件法律保护组织所发起的诉讼就属于这一类。由这类组织发起的诉讼并不特别活跃，这与开源法律保护组织的性质和资金有关：开源法律保护组织具有公益性，其提供的法律服务（包括诉讼）都属于pro-bono的性质；它们的资金来源于（会员）捐助，诉讼目的往往源于开源组织的委托。

虽然这类诉讼并不频繁，但具有较高的成功率及很好的先例作用和示范效应。例如SFLC针对使用Busybox的消费类电子厂商提起的诉讼，多数都取得了诉讼胜利、和解等较好的结果。这类诉讼能对破坏开源生态环境的行为产生某种震慑作用。同时，软件行业还出现了检查开源软件代码的工具（如Blackduck等），也给这类诉讼提供了更多的线索和机会。当然，这些诉讼尚不能完全解决开源合规性问题，这些问题绝大多数并非通过法律手段，就在同开源社区的交流和合作中解决了，因为同开源社区的合作关系往往是更为重要的。

应当考虑：使用开源软件是否具有法律风险，是否符合开源生态环境要求。

我应该怎么做——使用开源软件/代码须符合许可协议

目前开源社区已批准了60余种开源软件许可协议。如上所述，软件许可协议是基于软件享有的著作权保护。即使开源软件或代码是免费获得的，也并不意味着程序员可以自由或随意使用，对开源软件的使用（复制、分发、修改等）必须获得著作权人明示的许可，并按照开源软件许可协议规定的方式和要求使用。

开源软件许可协议的主要特征一般有三方面：首先，源代码可以获得并修改；其次，二次分发不收费；第三点是最为特殊的——要求将修改后的代码开放给开源社区。如果有第三点要求，那么就属于Copyleft类型的协议，否则属于Permissive类型的协议（见第四部分对两类开源许可协议的介绍及图1）。

另外，开源软件许可协议对使用者还提出哪些要求或规定呢？比如，要求转载版权声明和许可协议，认可原作者；要求继续往下游提供、传播源代码，将修改的代码开放给社区；甚至将与修改代码相关的专利免费授权给公众使用等。

应当了解：所选用的开源软件或代码，其许可协议是哪种，对使用者有什么要求。

我应该了解什么——重要的许可协议

开源软件许可协议主要分为两大类（见图1）：第一类是互惠的许可协议或称“Copyleft”，它要求将修改的代码开源，回馈给开源社区；第二类是更为包容“Permissive”的许可协议，它允许修改的代码不被开源。

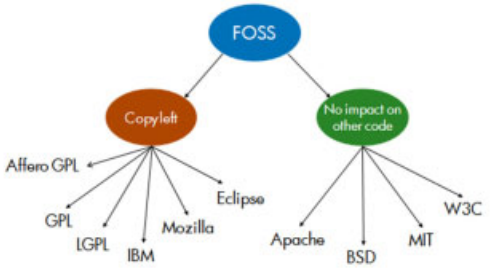


图1 两类开源软件许可协议

■ 包容性（Permissive）的许可协议（如Apache、BSD、MIT、W3C等）对使用者的要求和限制极少，但这类软件仍不能被认为处于“公有领域”（Public Domain）而可以随意使用。它一般要求使用者须转载该软件的版权声明和许可协议，但对于修改源代码及二次分发并无限制。

■ “Copyleft”许可协议要求使用者（当分发该开源软件的二进制代码时须）继续向下游传播其源代码，而且要求修改的源代码必须公布。典型的协议有GPL、LGPL、AGPL、MPL和EPL等。在这类许可协议中，各协议的严格程度各不相同，修改源代码产生影响的范围亦不同。

GPL协议是最著名的开源软件协议，是Copyleft类型许可协议的代表，已在2007年从版本2升级到版本3。与GPL v.2相比，GPL v.3更为严格，它对代码修改者/贡献者所拥有的相关专利有更大的影响。如采用GPL v.3协议的嵌入式软件须使最终用户能对该嵌入式软件的代码进行修改等。

Affero GPL协议是为了克服GPL协议的漏洞而产生的。网络服务商在服务器端使用GPL开源软件，但并不发布软件产品，由于此时其没有对该开源软件或代码进行“分发”（distribute），则可以自由地使用GPL软件却无须提供源代码或修改后的代码。在改进后的Affero GPL协议下，只要用户通过计算机网络与所谓的“程序”有交互，无论是否复制、分发、传播，都须将该程序（包括其修改或衍生作品）开源。目前，采用AGPL的开源项目在不断增长，截至2012年6月已经有413个开源项目采用AGPLv3协议。

应当关注：开源软件许可协议的类型（Copyleft/Permissive），对代码进行修改或二次分发是否需要公布源代码，对非开源代码的影响程度有多大。

中国的知识产权保护与开源软件合规

对中国的软件行业来说，采用开源软件和代码无疑是一条迅速发展的路径，但采用开源资源时，我们不应以破坏开源“生态环境（系统）”为代价。知识产权保护在中国非常具有挑战性，即使违反开源软件许可协议也不一定招致法律诉讼或导致严重后果，开源法律保护组织的维权活动目前也尚未延展到亚太地区。但如上所述，任何一个开源项目均需要维护其开源“生态环境（系统）”，不尊重开源“生态环境（系统）”的行为无疑将受到开源社区的排斥，使用开源软件是否合规将直接影响与开源社区的合作关系。作为中国的程序员，对开源协议是否了解、使用开源软件是否合规、是否尊重开源“生态环境（系统）”，都将影响中国在全世界的开源社区面前的形象。**这种意识能够帮助我们赢得世界（开源界和开源社区）的尊重。**

综上，本文介绍了使用开源软件应当了解的法律问题，概括为以下“五个应当”。

■ 应当明确：不论其收费与否，使用开源软件均须满足其许可协议的要求。

■ 应当考虑：使用开源软件是否具有法律风险，是否符合开源生态环境要求。

■ 应当了解：所选用的开源软件或代码，其许可协议是哪种，对使用者有什么要求。

■ 应当关注：开源软件许可协议的类型，对代码进行修改或二次分发是否需要公布源代码，对非开源代码的影响程度有多大。

■ 中国技术人员应当：注意合规地使用开源软件，这种意识才能帮我们赢得世界（开源界和开源社区的尊重）。

董颖

中国惠普有限公司高级知识产权律师，曾任通用电气（中国）医疗集团知识产权律师，负责中国区知识产权工作。

OceanBase：淘宝开源海量数据库

文 / 李震

世界上充斥着各种各样的轮子，这句话在IT技术界有特定的意义，我们用重复造轮子来形容那些投入大量时间、精力和金钱实现已有技术方案可以提供的功能的行为。在大量开源NoSQL系统发展得如火如荼，以Oracle、MySQL为代表的传统关系型数据库正在并将要发挥重要作用的今天，我们为什么需要一个名叫OceanBase的软件系统来完成看起来并不复杂的业务支撑呢？这要从互联网企业的数据应用说起。

互联网企业的数据由互联网用户产生，随着业务的扩大正比增长，并在网站发展到某个阶段开始以指数级别不断膨胀，这些数据是企业最核心的竞争力，承载和利用这些数据需要强力的数据架构。传统的集中式关系型数据库已接近了纵向扩展的极限，同时在成本考量上也处于劣势。新兴的分布式数据库通常需要在业务上做出比较大的妥协，并丧失一些扩展或变化的灵活性。OceanBase采用了一些新的技术方法尝试部分解决这个问题，针对的应用数据规模在千亿条记录，提供OLTP级别关系型数据库的基本功能。简单来说，OceanBase希望在这样的数据规模下，为业务提供类似于关系型数据库使用体验的在线数据服务，并且在成本、性能、功能三个维度上达到应用可以接受的平衡。那么，这种平衡能够体现在淘宝的哪些应用场景下呢？

我们分析了淘宝的大量系统，从核心的交易相关到类SNS及数据分析，涵盖了互联网的大部分应用类型，这些应用有一个共性，短时间内变化的数据总是远小于随着时间积累下来的数据，通常有两到三个，甚至更多的数量级差距，而大部分数据架构的复杂性，主要体现在对短时间内变化数据的处理上。这个特点使我们使用了一种新的思路来搭建OceanBase，这个思路也决定了OceanBase的应用场景，即使用者希望利用关系数据库的一些特点，例如联表、事务，同时希望不用关注数据膨胀带来的运维压力，短时间内数据变化不特别剧烈，对访问性能有较高要求。针对淘宝乃至其他互联网企业来讲，这个应用场景基本上包括了绝大部分的线上应用。随着需求推动系统逐渐发展，OceanBase通过简单的分布计算也具备了小规模、百TB数据量级别的部分OLAP功能，因此，OceanBase和其他系统相比更加轻量级，通常也能提供更高的性能。下面具体描述一下OceanBase采用了何种技术架构来实现成本、性能和功能的平衡。

架构特点

首先，OceanBase将数据拆分为基准和修改增量两个部分。基准数据在一个业务周期内保持不做变更，所有的修改增量做集中处理，每次数据查询将按照应用需求决定是否需要合并最新的修改增量。如图1所示。

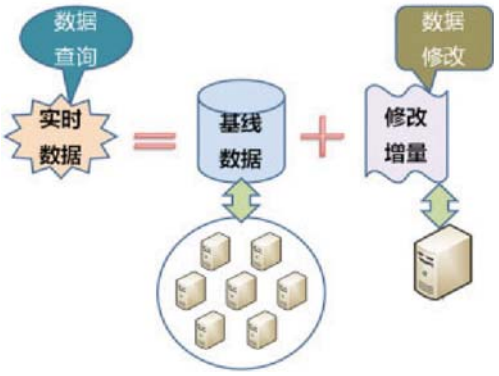


图1 数据分为基准和修改增量两部分

一个业务周期结束后，这段时间内的所有修改增量也将不允许修改，我们称这个过程为冻结。冻结后的修改增量数据将和基准数据合并为新的基准数据，如图2所示。

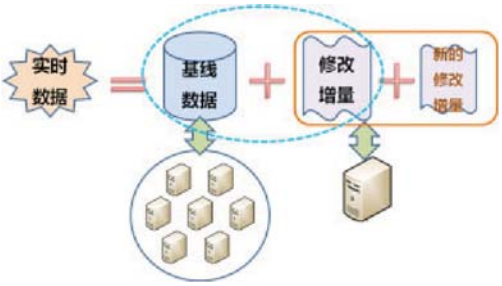


图2 冻结

从数据分布方面来看，OceanBase使用B+树作为主要数据结构，基准数据的B+树根节点集中管理，叶节点按照备份数配置分布在多台服务器上，同时保证全局有序。增量更新数据同样使用B+树提供存储和查询。

OceanBase通过这种数据模型，实现了一种分布式查询和集中式更新的思路，这种思路希望能够在一定的应用场景下，有效平衡性能、功能和成本（包括部署成本和开发成本），提供具有类似联表、事务等关系型数据库特性的分布式解决方案。那么，OceanBase的物理架构大概是怎样的呢?

功能模块

下面详细介绍一下OceanBase各个模块（如图3所示）所扮演的角色。

■ RootServer是一个轻量级的中心控制节点，负责接收所有节点的心跳信息，并维护基准数据B+树根节点信息以及基准数据分布服务器列表以备ApplicationServer查询获取。由于应用端存在必要的元数据缓存，RootServer短时间不在线通常不会对线上服务产生影响。

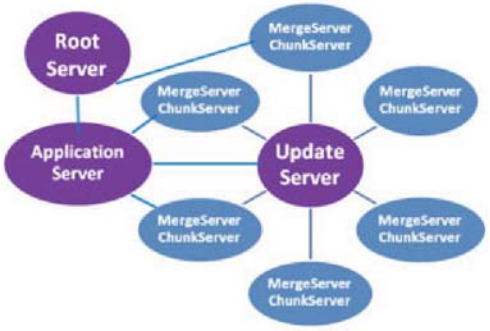


图3 OceasnBase各功能模块

■ ChunkServer负责存储基准数据B+树的叶节点，提供访问本地基准数据的接口，并负责本地基准数据与增量更新数据合并为新基准数据的过程。ChunkServer可以线性扩展，同时每个叶节点都有多个不同的ChunkServer存储，小规模的ChunkServer宕机通常不会对线上服务产生影响。

ChunkServer使用SSTable的方式存储数据，这些数据可以属于一张表或多张表，它们按照某些业务逻辑组成的主键连续存放在一起，这种方式类似于关系型数据库的聚集索引，在查询时可以尽量少的I/O操作获取数据。同时，OceanBase提供了将少量字段的数据集中存放的方法，称之为Column Group。这些Column Group可以由应用定义，在某种程度下给应用提供了平衡性能和灵活性的手段。

■ MergeServer负责处理应用查询请求，合并基准数据与增量更新数据，使应用获得最新的数据，并根据应用需求进行处理，如排序、过滤、统计等。MergeSever可以线性扩展，小规模的MergeSever宕机同样不会对线上服务产生影响。

■ UpdateServer是整个系统架构中最核心的节点，负责存储增量更新数据，维护内存中的B+树，提供数据更新与查询接口，并处理更新数据的转储与冻结过程。UpdateServer现在可以看作OceanBase中的一个单点，它是上帝打开的那扇门，集中式更新使我们可以很方便地实现关系型

数据库的部分特性。同时它也是上帝关上的那扇窗，为了消除这个单点带来的影响，我们做了很多努力和尝试。

为了避免这个单点UpdateServer带来的查询瓶颈，我们在UpdateServer上实现了读写分离，读取数据的能力做到了线性扩充。

为了避免UpdateServer单点故障给应用带来数据不可用的风险，我们在UpdateServer使用HeartBeat技术实现了HA，一个虚拟IP在两个或者多个UpdateServer节点之间漂移，降低UpdateServer单点故障的影响。

为了消除UpdateServer内存大小对更新数据量的限制，我们实现了数据转储机制，OceanBase在增量更新数据达到一定量时会启动后台工作将UpdateServer内存中的数据以SSTable的形式持久化到SSD磁盘上，将这个过程称之为转储。通过转储，我们可以在保障一定性能的前提下大大提高集群支持的增量更新数据量。

为了保证多台UpdateServer之间的数据一致性，我们使用了CommitLog的技术，OceanBase保证多个UpdateServer节点同步完成才会返回应用成功的信息。同时为了提升写入效率，OceanBase采用了一个相对复杂的group commit方式保证一批数据同时刷新磁盘，降低磁盘I/O。

通过上面一系列的技术手段，我们基本上解决了现阶段应用场景下UpdateServer单点带来的各种问题。相对于这个单点带来的各种好处，我们认为这种代价还是非常值得的。

优势VS.劣势

文章开头时提到轮子的问题，那么OceanBase和其他轮子相比有哪些优势和劣势，我们该怎样定位OceanBase呢?

首先，OceanBase实现了更新数据和基准数据的分而治之。通常说来，分布式NoSQL系统的最复杂部分在于更新数据的一致性，也由此产生了各种不同的一致性等级和解决方案。而大部分流行的开源NoSQL系统为了降低更新数据带来的随机写入操作，一般会采用只做追加操作并按需整理的方式处理，但整理的频度和力度直接影响访问效率。在各种系统调优中，数据整理的优化也一直是重点，只能根据具体业务分析，且很难能够达到最优。而且这些系统的节点需要考虑读和写两种功能，Memtable何时刷入磁盘、写入Buff大小如何配置等这些问题都会考验系统的侧重方向和运维的技术实力。

OceanBase尝试采用另一种思路绕过这些问题，将持久化数据和更新数据完全分离，用不同的硬件架构、不同的软件架构进行支持，也部分避免了这些问题的影响。相对来说，这两个架构可以各自发展，具有更加灵活的纵向扩展和横向扩展。同时利用业务低峰期批量进行类似整理的合并操作，也降低了运维难度，性能表现从业务方角度来看也更加平稳。

其次，随着SSD设备的不断发展成熟，所有的系统都面临着架构的冲击，通常来说，SSD设备在小块数据随机写入方面支持得不太好，这与擦除算法和写入放大有关。而细数一下OceanBase系统中的数据写入磁盘，一共有三个操作：一是Commit Log通过GroupCommit方式顺序追加写入；二是增量更新数据按照MB单位大小写入磁盘；三是增量更新数据与基准数据合并之后，生成新的基准数据并以MB单位大小写入磁盘。也就是说OceanBase在顺序大块写入的情况下，天然适应SSD设备，可以充分利用SSD设备比传统机械磁盘高出两到三个数量级的随机读取性能。

再次，应用仍然需要一些关系型数据库特性，例如联表和事务。OceanBase通过冗余表的形式部分支持联表功能，这种实现方法最大的技术挑战在于一条被联表的数据变更可能引起冗余表的多行数据变更，这种变更放大的效果有时甚至可能达到百万乃至千万倍。但OceanBase将更新数据放入内存或者SSD设备这类更快介质的特点使我们有了解决这个问题的办法，所有的更新都只存在一条，数据的联表操作发生在每次查询时，直到新的基准数据生成，冗余表被更新。而UpdateServer的单点更新也部分支持了事务，通过顺序执行以及B+树的copy on write等机制，OceanBase能够保证所有在一次请求中的操作事务性地被执行。

当然，OceanBase的劣势也显而易见，系统中的UpdateServer单点，使整个系统的可扩展性和性能都有了一个看得到的天花板，尤其是数据更新。另外，更加丰富的功能使接口更加复杂，用户的使用难度会随之提高。同时，由于是一个新开发的系统，周边的各种运维和监控工具还比较欠缺。OceanBase的定位是一个分布式关系型数据库，这三点是我们接下来最重要的改进方向。

发展方向

在扩展性方面，OceanBase的ChunkServer和MergeServer本身就支持线性扩充，当前UpdateServer支持了读写分离，读性能基本上也能够做到水平线性扩充。整个系统的可扩展性主要集中在数据写入量上。UpdateServer支持增量更新数据转储机制，但转储后的数据访问性能比在内存中有下降，并非最佳解决方案。在接下来的一段时间内，OceanBase会实现增量更新数据按需分发至ChunkServer，以内存缓存方式存在，这样更新数据量能够随着集群规模扩充而线性增长，缓解单点写入瓶颈。单位时间内写入量过大也会形成瓶颈，在应用需求达到瓶颈的情况下OceanBase会实现内部分库写入，形成一个达到相当规模的服务器集群。

性能提升相当于集群能力的纵向扩展，OceanBase后续在这方面的主要工作包括新的数据序列化方式、新的缓存机制、新的网络通信框架、哈希索引支持等。当然这些工作是有优先级的，现在也已有少量工作在做了。

在易用性方面，OceanBase能支持大部分的SQL语言集，并通过与PostgreSQL和MySQL客户端绑定的方式降低应用开发难度，提供更加丰富的访问语义。这项工作已经在进行中。

运维方面，OceanBase现在由一个专门的小组负责，有1~2个人管理线上接近200台服务器，还有2个人负责开发运维工具，已取得了不错的成果。

案例

OceanBase具有鲜明的特点，相对NoSQL系统经典的CAP理论来说，它更偏重于C（OceanBase提供最高的数据一致性）和A（OceanBase承诺5个9的系统可用性），兼顾P（基准数据提供良好的分区容错性）。同时，OceanBase也给应用提供了类似于关系型数据库的功能接口，方便应用在不大量修改业务逻辑的前提下进行数据方案替换，并取得了良好的效果。

以一个典型的案例来说明，这个应用的数据原本存放在16×2台MySQL数据库服务器上，按照某种主要业务ID分表。业务逻辑每次查询平均获取500条结果，这些结果需要和另一个业务数据进行join操作，然后对join后的结果进行排序或过滤操作，用户操作之后数据的更新要及时体现在页面上，需要非常高的数据一致性。原有的数据库随着数据量的逐渐增加，索引量越来越大，查询性能不断降低，已到达了瓶颈。而其他NoSQL系统没有这种相对复杂的功能，业务方也很难对逻辑进行改变。在这个场景下，使用OceanBase非常合适，最终使用12×2的集群（成本和原有集群基本相当）支撑了3~5倍的业务规模，同时也给应用方打好了业务发展的数据基础。目前这个应用单表数据已经突破80亿条，整个数据库数据超过了200亿条。

总结

回到文章开头的问题，我们不是在重复造轮子，在淘宝内部已有接近20个应用在使用OceanBase，大部分场景是使用其他数据架构无法解决或者很难用低成本解决的。我们希望这个轮子能够让淘宝的一些应用在特定的道路上跑得更好，并逐步通用起来，在合适的时间支持更多核心的应用。

最后，欢迎大家访问我们的OceanBase开源主页oceanbase.taobao.org，也可以加入邮件列表oceanbase@code.taobao.org。

李震

淘宝花名楚材。现任职于阿里巴巴集团技术平台部核心系统研发存储组，负责产品线包括TFS（淘宝分布式文件系统）、TAIR（淘宝KV存储系统）以及OceanBase（淘宝分布式数据库系统）。

NiuTrans：开源统计机器翻译系统

文 / 肖桐，张浩，李强，朱靖波

什么是NiuTrans

语言是人类表达思想和传播信息的重要途径之一。为了实现不同语言之间的无障碍交流，人们一直梦想着利用某种工具完成自动翻译，在这种背景下，机器翻译应运而生。发展至今，机器翻译已被真实地应用于人类生活的某些方面，具有一定的实用性。例如Google的在线翻译系统就是一个很好的例子，它提供了一种免费、方便的自动翻译接口，并展示了巨大的应用价值。在此基础上，我们提出了一个更加大胆的想法：是否可以根据不同的需求，由用户自己构建个性化的翻译系统？NiuTrans开源统计机器翻译系统使这个梦想成为了可能。

NiuTrans是东北大学自然语言处理实验室花费四年时间独立研发的一套开源统计机器翻译系统。Niu隐含了三种含义：做事要有勤勤恳恳的老黄“牛”精神、做研究要有敢于开拓创“新”（New）的精神，以及我们做研究的所在地——东北大学（NEU）。

NiuTrans系统的最大特点是所有翻译过程的构建完全由NiuTrans自动完成，它不依赖人工书写的规则和翻译模板，唯一需要的是一定量的双语互译的平行双语句对数据。实现这个全自动翻译过程面临着一个巨大的挑战——如何让机器能够从双语句对数据中自动学习翻译知识，并利用翻译知识对新句子进行自动翻译。

为了解决这个问题，NiuTrans使用了当今最先进的统计机器翻译技术，其基本思想就是把翻译问题抽象为一个求解最优翻译路径的数学问题。通过翻译模型的训练学习、语言模型的训练学习和翻译特征权重的训练学习等来完成整个机器翻译过程，最终利用当今计算机强大的运算能力，从海量的翻译候选集合中搜索到“最优”的翻译结果（这个过程通常也被称作解码）。

相比同类开源统计机器翻译系统，NiuTrans系统的另一个特点是它支持当今五个主流翻译模型，包括基于短语的模型、基于层次短语的模型、树到串模型、串到树模型和树到树模型。这些模型在NiuTrans中都使用统一框架来实现，提高了代码的重用性，降低了开发和升级的难度。此外，作为一个开源项目，我们在设计NiuTrans之初就希望更多的机器翻译爱好者参与NiuTrans的发展与完善，为此我们提供了非常丰富的优化算法和API。这么做一方面可以供研究开发人员使用NiuTrans时进行系统调优，另一方面也便于对NiuTrans系统的修改和二次开发。

早在2007年，英国爱丁堡大学就曾发布了开源统计机器翻译系统Moses，目前被学术界及工业界广泛使用。其实国内的一些研究机构也尝试开发过开源机器翻译系统（如丝路），但由于当时技术不成熟及系统后期维护和更新的停滞，机器翻译系统开源工作在国内进展始终非常缓慢。2007年底，我们实验室启动了NiuTrans项目，当时实验室主任朱靖波教授说：“NiuTrans的研制目的是让世界上每一个人都能拥有属于自己的个性化机器翻译系统。”本着这个信念，NiuTrans开发团队使用了当今最先进的翻译模型（如基于句法的翻译模型），而且不断对技术进行升级，在2011年7月初推出第一个版本后，终于在2012年7月推出了包含五套翻译模型的NiuTrans完整版。与其他开源系统相比，NiuTrans系统能够支持更广泛的翻译模型和更快的翻译速度，特别是汉语到外文翻译性能出众。凭

借这些优点，在2011年7月开放源代码至今的短短一年内，NiuTrans已被500多个国内外的个人和研究机构注册下载使用，并在CWMT11及NTCIR等机器翻译评测中取得了多项第一和第二的好成绩。特别是2012年7月，在韩国举行的计算语言领域最顶级盛会ACL会议上，NiuTrans系统进行了系统演示，引起了包括美国约翰霍普金斯大学、英国爱丁堡大学等许多国际知名机器翻译研究机构及工业界同行的广泛关注。

NiuTrans架构及技术特点

NiuTrans系统是一套基于统计模型的机器翻译平台。统计机器翻译本质上将翻译任务看成一个数学模型最优化问题，其形式化的定义如下：

$$t^* = \arg\max_t \Pr(t \mid s)$$

其中，s表示一个源语言句子，t表示一个目标语句子，Pr(t|s)表示翻译模型将源语言句子s翻译成目标语句子t的概率，t^*表示使翻译概率Pr(t|s)达到最大的目标语句子。统计机器翻译模型需要解决以下三个基本问题。

- **建模问题**：如何定义翻译概率Pr(t|s)。
- **训练问题**：如何从数据中学习模型参数。
- **搜索问题**：如何在给定翻译模型条件下寻找到最优目标语译文。

对于第一个问题，研究人员往往根据独立性假设将翻译概率Pr(t|s)进一步分解展开成更小翻译单元上的概率积。在这种情况下，机器翻译的过程可以被描述成以下三个步骤：

- 将源语言句子分割成包含若干最小翻译单元的序列；
- 按一定策略选择并翻译源语言句子中的最小翻译单元；
- 组合最小翻译单元的译文，并选择模型代价最低的组合结果作为整个句子的译文。

不同的统计机器翻译模型对最小翻译单元的定义不同。例如，基于短语的统计机器翻译模型的最小翻译单元是连续的词串，基于句法的统计机器翻译模型的最小翻译单元是子树片段。对于第二个问题，NiuTrans的训练模块可以自动地从双语句对数据中学习翻译规则并优化模型参数。对于第三个问题，NiuTrans平台的解码模块可以快速高效地完成对最优目标语译文的搜索，完成翻译的整个过程。

NiuTrans系统架构如图1所示，其中深色圆角矩形代表数据资源，浅色方角矩形代表功能模块。

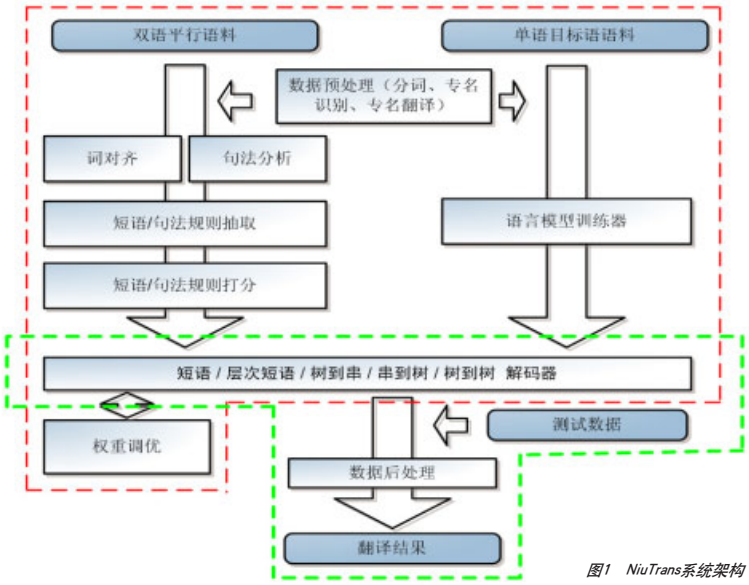


图1 NiuTrans系统架构

NiuTrans所有模块分属于训练和解码两个阶段。其中用红色虚线框所包含的功能模块属于训练阶段，用绿色虚线框所包含的功能模块属于解码阶段。在训练阶段，系统从训练数据中学习得到模型和模型参数。每个功能模块的作用如下所述。

- 数据预处理模块完成对训练数据集的预加工，包括分词、词性标注、专名识别和专名翻译等。
- 词对齐模块为双语平行数据中的每一句对抽取得到词对齐信息，使得互为译文的源语词汇与目标语词汇联系起来。
- 句法分析模块用于分析源语和目标语句子，得到其对应的基于短语成分的句法分析树。句法分析树是训练基于句法的统计机器翻译模型的一个必要输入。
- 短语抽取模块用于从包含词对齐信息的双语平行语料中抽取短语翻译规则。
- 句法规则抽取模块用于从带有句法分析树和词对齐信息的双语平行语料中抽取获得句法翻译规则。

■ 规则打分模块用于对每一翻译规则进行概率估计和打分，如翻译概率、语言模型概率、词数惩罚特征、规则数惩罚特征等。

■ 语言模型训练模块用于从目标语单语语料中自动学习得到语言模型。语言模型是用来评价目标语译文的流畅度的一个特征函数。

■ 权重调优模块用于在开发数据集上对翻译模型特征权重向量进行调优。

在解码阶段，所包含的功能模块的作用可以描述如下。

■ 解码器是整个NiuTrans最重要也是对翻译性能影响最直接的一个模块。该模块主要完成从搜索空间中找出最佳目标语译文，即完成真正翻译的过程。

■ 数据后处理模块对解码器产生的目标语译文做进一步优化，使其更符合书写和阅读习惯。

NiuTrans在设计与实现时不仅考虑了对现有算法的优化，同时为了优化系统的运行时间和存储空间，也考虑了一些工程上的实用技术。所使用到的主要技术可归纳如下。

■ 支持的统计机器翻译模型非常广泛。目前统计机器翻译模型可以分为三大类（如图2所示），包括基于短语的翻译模型、基于层次短语的翻译模型和基于句法的翻译模型。NiuTrans已能够支持上述三类统计机器翻译模型。

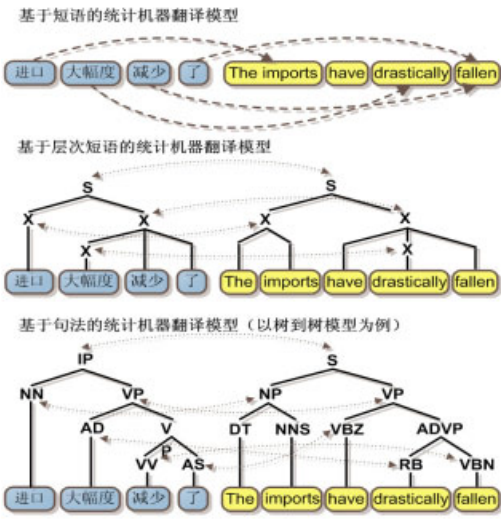


图2 NiuTrans支持的统计机器翻译模型

■ 所有翻译模型在NiuTrans中使用同一个系统架构实现。这大大减少了对公用模块的重复开发，使得整个系统更加简洁和易于维护。

■ 规则抽取模块考虑了一些启发式算法和参数设置，允许用户在抽取规则时对算法做一些约束，从而减少抽取出来的规则数量，并且不对翻译性能产生太大影响。

■ 语言模型训练模块引入了低频限制策略，可以有效减少噪声数据的影响，降低语言模型的大小。生成的语言模型以trie数据结构存储为二进制文件，可进一步降低语言模型所需的硬盘和内存空间。在解码器中计算语言模型概率时，查询算法中集成了Cache机制，大大提高查询速度。

■ 统计机器翻译对目标语最佳译文的搜索过程被证明是一个NP难题。为了解决这个问题，NiuTrans解码器在使用CYK算法的基础上，引入了一些剪枝技术，如束搜索技术和立方剪枝技术等，从而快速高效地寻找到最优或近似解。

■ 解码器中可同时使用多个调序模型，有效缓解了基于短语的翻译模型对调序能力的不足，最终提高了机器翻译性能。

■ 解码器支持多线程技术，可以有效利用CPU资源，加快解码速度。

■ 整个NiuTrans平台使用C/C++语言编写完成，不依赖于任何外部开发的开源代码。

■ NiuTrans提供了一些应用开发接口，允许用户添加一些自定义翻译特征，具有良好的扩展性。

NiuTrans面向的用户非常广泛，不仅包涵专业的研究人员和开发人员，还包涵非专业的普通用户。为了使用NiuTrans构建机器翻译系统，用户需要准备一定规模的双语数据。由于NiuTrans是一套基于统计模型的翻译系统，原理上数据规模越大、数据质量越高则翻译性能越好。但对于数据规模和质量的要求需要根据具体应用环境而定。例如，如果用户想构建一套通用领域的翻译系统，那么由于数据范围非常宽泛，所需要的数据规模可能需要达到上千万的句对，并且对数据质量的要求会比较高。如果用户只想构建一套针对特定产品的说明书的翻译系统，那么所需的数据规模就不需要太

大。使用NiuTrans做研究的研究人员不仅需要准备自己的实验数据，还需要对NiuTrans的系统参数和高级用法有更深入的理解。这是由于系统参数的设置对实验结果也会产生一定的影响。在NiuTrans基础之上作二次开发的开发人员则需要了解系统的整体架构和每一基本模块的输入输出，并掌握NiuTrans所提供的程序接口和文件的数据格式。

NiuTrans在设计之时就考虑了平台可移植性问题。目前，NiuTrans支持Windows和Linux操作系统的32位和64位模式。对于系统部署节点的硬件环境，NiuTrans没有一个严格的限制。最低硬件环境的配置需要根据用户应用NiuTrans时所使用的数据规模而定。一般情况下，硬件设施不要低于4GB内存和20GB硬盘空间即可。对于系统部署节点的软件环境，NiuTrans在编译时只要求有基本的编译器（例如，Windows下的Visual Studio 2008和Linux下的gcc/g++）；在运行时只要求对Perl脚本的运行提供支持，并且在Windows环境下安装了cygwin应用程序。

应用场景

在线文本翻译：NiuTrans系统的最直接的一个应用是在线文本翻译。随着互联网的普及与发展，数以亿计的网民获取全球信息的期望值越来越高，在线文本翻译为这一需求提供了一个合理的解决方案。与传统特定领域的翻译任务相比，在线文本翻译的最大特点是互联网用户输入文本的多样性及不规范性，这就要求翻译引擎能够适应不同类型的文本翻译需求，如新闻翻译、博客翻译、评论翻译等。相比传统的基于规则和模板的翻译系统，NiuTrans的统计机器翻译引擎中采用了多种剪枝及优化技术，具有很高的翻译速度，这一特点也使之能够处理互联网用户海量数据的翻译请求。

语音翻译：NiuTrans系统可作为语音翻译中机器翻译组件的核心翻译引擎。随着全球一体化进程的加快，不同国家的人之间的交流变得越来越频繁，语音翻译为语言无交集的人群提供了流畅交流的可能。语音翻译的最大特点是通过语音输入，要求翻译实时性与准确性，这就要求翻译引擎能够具有较快的处理速度与较强的容错能力。使用语音翻译技术，可以真正达到沟通无国界的目的，地球可真正称为“地球村”。

计算机辅助翻译：NiuTrans的另一个应用是计算机辅助翻译（CAT）。CAT是指在人工翻译过程中辅助使用计算机程序的自动翻译功能。CAT技术的翻译过程为，首先由机器翻译系统生成一个初始的翻译结果，之后用户在翻译结果的基础上指定几个基本正确翻译的单词或词组，然后机器翻译系统在此基础上继续进行翻译，该过程不断迭代下去，直到生成优秀的译文。CAT技术最重要的特点是人机交互功能，NiuTrans在开发的过程中为人机交互功能留有接口，仅需按照NiuTrans的相关规范进行修改即可。CAT技术可使得繁重的手工翻译流程自动化，大幅度提高翻译效率与翻译质量，帮助翻译工作者优质、高效、轻松地完成翻译工作。

多语言信息集成：NiuTrans可作为多语言信息集成技术的核心翻译引擎。在包含海量数据的互联网中，用户如何获取最想得到的信息成为一个挑战。传统的搜索引擎中，用户输入一种语言的检索词后，仅可检索到包含输入词串的检索结果。在这种条件下，用户查询自己关心的内容时并没有在整个互联网数据中进行查询，只是在包含自己检索词使用语言的单语数据子集中进行查询。如果用户的检索词在进行检索前可转换成不同语言的相应的检索词，那么用户可获得更加完整的信息，多语言信息集成技术可以使得对互联网资源的利用最大化。

NiuTrans研发团队

东北大学自然语言处理实验室（http://www.nlplab.com）由姚天顺教授创立于1980年，是国内最早从事机器翻译研究的实验室之一。目前实验室在朱靖波教授领导下，从事多国语机器翻译、语言分析、文本分析和语言工程等领域的研究工作，在学术研究和学术论文发表等方面取得了累累硕果。

Jscex：回归JavaScript的异步流程控制类库

文 / 赵劼

Jscex是迄今为止我最为投入的开源项目，已经坚持了两年，并用许多精力去推广。我做过很多所谓的“开源”，有些可能是随手放出一些自己常用的代码，还有一些可能是相对更为严谨的项目，但与Jscex相比，它们基本上都是消遣用的玩具。对许多程序员来说，写代码可以算作是一件颇为有趣的事情，但要将一个项目真正推广出去，便需要在更多方面做出努力，例如文档、示例、各种场合下的演讲等。拿Jscex来说，非技术领域方面的投入花费了我更多精力。这些事情有时会让开源作者烦躁，但我认为这也是对项目的考验，倘若有所松懈，会影响到项目的其他方面，包括项目本身的质量。

浅尝辄止

JavaScript是一门单线程语言，用户代码一旦发生阻塞则整个程序便会陷入停滞，因此它天生带有很强的异步特性，例如浏览器端的AJAX请求便是最著名的异步操作之一，此外如setTimeout也是一个异步方法，它会将用户传入的函数延迟一定时间后执行：

```
setTimeout(fn, 1000); // 1000毫秒后执行 fn 函数
```

异步编程其实十分困难，因为程序员早已习惯了线性的表达算法，但异步回调会将逻辑拆分得支离破碎，让我们难以对异步操作进行组合及协作，而异步操作的取消及错误处理也是一大难题。我最喜欢用“排序算法动画”来说明这个问题，因为它可以让我们很清楚地观察到一段十分简单的逻辑是如何被一个简单的异步操作破坏殆尽的：

```
var compare = function (x, y) {
    return x - y;
}

var swap = function (array, i, j) {
    var t = array[i]; array[i] = array[j]; array[j] = t;
}

var bubbleSort = function (array) {
    for (var i = 0; i < array.length; i++) {
        for (var j = 0; j < array.length - i; j++) {
            if (compare(array[j], array[j + 1]) > 0) {
                swap(array, j, j + 1);
            }
        }
    }
}
```

以上是一段非常经典的冒泡排序算法：使用双重循环遍历数组元素，一旦发现相邻元素的顺序不对，就进行交换，这样最大（或最小）的元素便会如水中的气泡般不断向上冒，直至整个数组排序完毕。那么我们又该如何用动画来演示算法的排序过程呢？

理论上说，既然已经有了排序算法，那么制作一个动画的思路十分简单：

- 让“比较”和“交换”耗费一定时间（决定排序快慢的主要操作）；
- 在交换元素后重新绘制数组。

如果我们在C#或Java这样的平台下实现这一功能，只需要开启一个额外的线程，并在合适的时候调用线程的sleep方法即可。不过，我们在JavaScript中又如何能让代码“暂停”一段时间呢？答案很简单：做不到。JavaScript是一门单线程语言，所有的代码都是在GUI线程上执行，我们

不能“阻塞”任意一段代码，否则界面就会失去响应，更别提“重绘”了。在JavaScript中唯一让代码“等待”的手段只有之前提到的setTimeout函数：

```
var compare = function (x, y) {
    setTimeout(function () {
        return x - y;
    }, 50);
}
```

我不止一次看到类似的代码，一般都是初次接触JavaScript异步编程（或是其他语言中基于回调的异步编程模型）的同学问我这段代码为什么无法正常工作。这其实是显而易见的，因为setTimeout会永远立即返回，它并不能让compare方法等待50毫秒，于是此时的compare将会立刻返回undefined而不是x-y，至于x-y则会孤零零地飘荡在JavaScript的执行队列里，50毫秒后执行，但已和原本的compare方法毫无关联了。

setTimeout正确的使用方式应该是这样的：

```
var compareAsync = function (x, y, callback) {
    setTimeout(function () {
        callback(x - y);
    }, 50)
}
```

为了用上setTimeout这样的异步方法，我们也必须让compare方法异步化，因此这里便使用compareAsync以示区分。我们让它多接受一个callback参数，并在setTimeout的回调函数内部执行，这种使用“回调函数”传递结果的方式是一种典型的异步操作模式，此所谓“Continuous Passing Style”。同理，swap方法也需要如此改变。此时你应该也已意识到了，一旦某个方法需要异步化，则所有直接或间接使用这个方法的代码都会受到影响。如果你尝试使用这种方式来实现冒泡排序算法的动画，就可能会得到非常糟糕的代码。

由于我们再也无法使用for或while这种最普通的方式来编写循环，就必须使用独立的方法来代替原本十分简单的双重循环，在“外层循环”方法中调用“内层循环”方法，接着在“内层循环”方法中进行“比较”和“交换”，然后再去调用“外层循环”方法。此时的逻辑流已不像原本的代码这样清晰明了，这便是异步编程最大的问题：即便编写不算困难，其最终结果也会将语义破坏殆尽。要知道一段代码的阅读次数往往会比编写和修改次数多得多，因此丧失语义会给代码的可维护性带来灾难。

几乎每个JavaScript程序员都想去改善异步编程的流程控制问题，再加上JavaScript的门槛实在太低，导致在Node.js上的相关类库有80余个之多，更别提那些只为浏览器设计的流程控制类库了。Jscex只是其中之一，如果使用Jscex来实现冒泡排序算法动画，则代码基本上是这样的：

```
var compareAsync = eval(Jscex.compile("async", function (x, y) {
    $await(Jscex.Async.sleep(10)); // 每次“比较”耗费10毫秒
    return x - y;
}));

var swapAsync = eval(Jscex.compile("async", function (a, i, j) {
    var t = a[i]; a[i] = a[j]; a[j] = t;
    repaint(a); // 交换后重绘
    $await(Jscex.Async.sleep(20)); // 每次“交换”耗费20毫秒
}));

var bubbleSortAsync = eval(Jscex.compile("async", function (array) {
    for (var i = 0; i < array.length; i++) {
        for (var j = 0; j < array.length - i; j++) {
            var r = $await(compareAsync(array[j], array[j + 1]));
            if (r > 0) $await(swapAsync(array, j, j + 1));
        }
    }
}));
```

对于初次接触Jscex的开发人员，我总是强烈建议忽视Jscex中无处不在的架子代码：eval(Jscex.compile("async", …))。尽管这段代码是标准的JavaScript代码，也在发挥它作为标准JavaScript代码的正常执行效果，但它在Jscex中的“语义”完全只是为了区分普通方法和一个Jscex方法。在实际应用中，这段代码也毫无变化，每次出现的形式均是如此，因此我建议开发人员将这段代码加入编辑器，一个快捷键便能直接输入。

排除这些架子代码，我们会发现如今的实现和原本的冒泡排序代码可谓如出一辙，唯一的区别只是在合适的地方增加了$await(Jscex.Async.sleep(…))操作而已。从语义上看，我们也能很明白地意识到这是一个传统的Sleep操作，代码将在这里“停留”片刻再继续下去。可以这么说：这段代码直接表达了我们之前的设计意图，并没有被由于异步回调破坏代码的语义表现能力。

设计思想

由于Jscex实现方式上的特殊性，许多人会对Jscex究竟是什么样的东西产生误解。例如，有些人会误认为Jscex是：

- **新语言**：Jscex虽然是对JavaScript语言“打了一个补丁”，但无论是从语言特性，代码语义乃至编程体验它都与JavaScript相同。因此，我每次都会强调“Jscex就是百分百的JavaScript语言”。
- **运行时**：Jscex与世界上绝大部分的JavaScript项目一样，都是普普通通的JavaScript组件，可以运行在任何JavaScript环境（浏览器、Node.js等）或是执行引擎（Rhino、V8等）上。
- **框架**：Jscex是百分百的类库，一种辅助你写代码的工具，而所谓“框架”是为开发人员留好填空的地方。“类库”的适应性一般会比“框架”好得多，就像你完全可以将Jscex与其他任何你喜欢的类库或是框架一起使用。

作为一个异步流程控制类库，Jscex的唯一目的便是“改善编程体验”，这跟其他大大小小异步流程控制方案有着相同的目标，但改善的“程度”以及改善的“方式”便是Jscex与它们最大的区别。在传统的异步流程控制类库中，人们普遍使用构建各类API的方式来辅助异步逻辑编写。例如，著名的Step类库提供了一个Step方法来辅助异步操作的顺序执行，为了支持“并行”和“分组”又分别提供了parallel和group方法，而功能更为丰富的Async类库则提供了十几种异步流程控制或数据处理的辅助方法。

这就产生了矛盾：更强大的类库，则往往意味着更多的API，以及更多的学习时间，这样开发者才能根据自身需求选择最合适的方法。此外，API的粒度也是一个课题。粒度越大的API往往功能越强，可以通过少量的调用完成大量工作，但粒度大往往意味着难以复用。越细粒度的API灵活度越高，可以通过有限的API组合出足够的灵活性，但付出的代价便是代码的“表现力”。

但其实说起“流程控制”，在JavaScript编程环境中，还会有比JavaScript语言本身更为合适的工具吗？JavaScript语言已提供了开发人员控制流程所需的全部关键字，而且开发人员已无数次证明了这些关键字是多么的灵活与优雅。假如，我们这里先说“假如”——假如我们可以使用传统的JavaScript语言编写异步代码，使用JavaScript来控制流程、表达逻辑，那么就可以避免学习大量新的API，避免遭受JavaScript中的语法噪音，只需编写简单而熟悉的JavaScript代码即可。使用JavaScript语言本身提供的特性进行流程控制，对于JavaScript开发人员来说，是一件天经地义、顺理成章的事情。

在软件开发行业有一句很著名的话：Every abstraction is leaky，即任何一种抽象都是有缺陷的。事实上，各种异步编程模型都是种抽象，它们是为了实现一些常用的异步编程模式而设计出来的一套有针对性的API。但在实际使用过程中我们可能遇到千变万化的问题，一旦遇到模型没有“正面应对”的场景，或触及这种模型的限制（如Step的this回调），开发人员往往只能使用一些相对较为丑陋的方式来“回避问题”。Jscex也是一种抽象，因此也不可避免地出现leaky，但Jscex的设计思路便是将这种抽象构建为JavaScript本身。换句话说，Jscex的能力在很大程度上是受限于JavaScript语言的表达能力，而传统异步编程类库则是受限于自身设计的API的表达能力。哪种做法可以应对更大范围的场景，我相信应该已不言而喻了。

Jscex选择的是一种方式：开发人员使用最普通的JavaScript来表达一段逻辑，只不过通过某种方式来指定某一个环节是“异步操作”，代码执行流程会在这里“暂停”，等待该异步操作结束，然后继续执行后续代码。如果这个操作失败，则相当于抛出了一个异常，会被catch语句捕获。

当然，单纯依靠JavaScript语言本身显然无法获得这样的能力，JavaScript的执行流程不会随便暂停，因此我们需要对代码进行自动改写。这便是Jscex的工作原理，值得注意的是“自动”二字，即开发者编写代码时完全不需要额外的“改写”步骤，这个功能在Jscex中叫做“JIT（Just in Time）编译”，也就是在程序的执行过程中动态生成并执行新的代码。

Jscex自动改写的最小单位是“函数”，传统

JavaScript函数是这样定义的：

```
var func = function () {
    // 函数体
};
```

而Jscex函数是这样定义的：

```
var jscexFunc = eval(Jscex.compile("async",
    function () {
        // Jscex 函数体
    }
));
```

在使用形式上，Jscex和普通类库毫无二致，这便保证了它与JavaScript有着相同的编程体验。CoffeeScript或Dart等以JavaScript作为目标语言的“新语言”，都必须引入一个额外的编译步骤。我反复强调，使用Jscex进行开发与普通的JavaScript编程几乎没有任何区别。

在Jscex函数中，我们可以使用唯一的语言扩展——“绑定”操作，这可以简单理解为那个异步的$await操作。“绑定”操作的目的是告诉编译器哪个地方需要“特殊对待”，例如以下这个Jscex函数：

```
eval(Jscex.compile("async", function () {
    var img = $await(readAsync("http://..."));
    console.log("loaded!");
    $await(writeAsync("./files/..."));
    console.log("saved!");
}))
```

经过编译器转化之后实际等价于：

```
(function () {
    var _b_ = Jscex.builders["async"];
    return _b_.Start(this,
        _b_.Delay(function () {
            _b_.Bind(readAsync(...), function (img) {
                console.log("loaded!");
                return _b_.Bind(writeAsync(...), function () {
                    console.log("saved!");
                    return _b_.Normal();
                });
            });
        })
    );
})
```

可见，每个$await之处都会被转化成Bind调用，从原本的“顺序”代码变成了“回调”形式，这保证了只有readAsync及writeAsync操作结束后才会继续执行后续代码。Jscex将会把一段标准的JavaScript函数编译成统一的Monad形式，并由注册在Jscex.builders集合上的“绑定器”负责实现特定效果。例如上述代码使用了async异步绑定器，我们还可能使用promise绑定器来支持Promise/A异步编程模型，以及seq绑定器来实现如Python和C#中的迭代生成器。

“绑定”操作使用了JavaScript的“函数调用”语义，这同样符合JavaScript的编程实践。有些朋友建议把$await改成关键字形式，而不是函数调用。可惜，这么做首先会被JavaScript运行环境拒之门外，导致无法实现JIT编译器，其次也有可能会破坏一些相关的事物，例如开发工具里的代码着色，甚至某些高级IDE中的语法提示，因为语法提示功能需要先对代码语义进行完整的分析。因此，保证Jscex与JavaScript绝对一致，其实也是在保证Jscex能够完美融入整个JavaScript生态环境。

不仅是语法设计上如此，Jscex自带的类库设计也充分借助了JavaScript的语义实现。例如Jscex的异步任务模型支持取消操作，要将一个任务置为“取消”状态，只需抛出一个未经捕获的CanceledError错误即可。根据JavaScript的语义，一个未捕获的异常会顺着调用链不断抛出，导致一系列的异步任务都会进入“取消”状态。这是一个统一而简单的取消模型。

总结

目前，Jscex在GitHub的关注人数已超过700人，在Node.js官方“异步流程控制”列表的80多个模块中排名第三，仅次于著名的Async（2600+）及Step（900+），还超过了著名的node-fiber（500+）。重要的是，Jscex只在国内进行过推广，而其他几个都是世界范围内的知名项目。不过我这里并非是想说明Jscex有多么成功，恰恰相反，我是想说明Jscex目前处于一个非常尴尬的境地。Jscex目前的社区参与度极低，例如Bug汇报数量，邮件列表的活跃度，乃至外部应用数量都与其关注度不相匹配。因此，提高社区参与度是我目前最需要解决的问题。

赵劼

网名老赵，目前担任IBM中国高级咨询师。互联网技术玩家，关注前沿技术，对编程之美、语言设计等相关问题有着浓厚的兴趣。目前专注于开源项目Jscex的开发与推广。

Coreseek：中文检索系统

文 / 李沫南

缘起

Coreseek是开源的全文检索系统Sphinx针对中文用户的改进版本。我进入全文检索领域纯属偶然。当时，Google刚刚上市，各种垂直搜索如雨后春笋，我觉得这是个挺有意思的方向。

为了实现中文的检索系统，中文切分的问题必须解决，当时流行的是张华平博士的ICTCLAS，可惜开源版本的ICTCLAS存在一些问题，很难在生产环节中使用。因而又不得不学习如何进行中文分词。之后，实现了一套基于条件随机场（CRF）的中文切分算法。

在2008年，为了卖CRF分词法，我又实现了一个基于复合最大匹配的分词算法MMSeg，并将MMSeg和我们对于Sphinx的改动，公布在Coreseek网站上。这就是现在几乎成为PHP网站全文检索解决方案标配的Coreseek的最初。（虽然Coreseek已经有很多商业支持的用户了，可惜CRF分词法确实一套也没卖出去。）

选型的历程

在初始阶段，因为我能力实在有限，从头写一套检索系统是连想都不敢想的事情——写不出来，找现有开源的，改！

选择在现有的开源软件基础上修改，而非重新开发还有另外一个原因。我的第一份工作是做一个.NET平台的ORM工具，在我做完之后，项目验收评审时，当时的领导突然问：“为什么你不直接使用NHibernate？你的系统有什么优势？”我当时哑口无言，不过现在看来，创造自己的ORM工具，除了满足了过分旺盛的创造欲之外，并没有额外地为社会创造多少价值。而我因此也养成了一个习惯，在决定新开发一个软件或功能模块之前，先去检索现有的全部可参考的设计和软件代码。这就如同读历史，读不同人物的悲欢离合，就如同自己经历了各种各样的人生一样；从现有的开源系统开始研究，会发现很多事前完全没有想到的问题，而把代码研究过一遍后，就已大体知道哪些地方需要特别留意了。

当时开源的检索系统已经有不少了，比如Lucene、Hyperestraier、Xapian、Sphinx、Lemur等。在原型系统选择之前，我先制定了一个选型的标准：

- 必须在“大”数据量下经过生产系统的实际考验过；
- 建立索引的速度必须足够快；
- 搜索的速度必须足够快，必须支持数据分布在多台机器；
- 索引文件大小要在一个合理的范围内；
- Rank机制要灵活。

Lucene：当时Nutch还根本不存在。而Lucene建立索引的速度尚可忍受，但当时不支持多机，另外加上Java语言导致Lucene的代码小文件特别多，超出了我能够理解的范围，因此Lucene被排除。

Hyperestraier：这个项目国内可能很少有人听说，但这个作者后来的作品，大家可能都知道——Tokyo Cabinet。在当时的评测中，虽然Hyperestraier不需要任何修改就可以支持中文（基于N-Gram），但建立索引的速度非常慢，索引文件非常大。

Xapian：代码非常清晰，倒排索引的存储方式和MySQL类似，支持多种存储模式，代码清晰相对好改。但在测试使用Xapian建立索引时，建立索引的耗时非常大，索引文件的大小是原始文本的几倍。Xapian有一个Sphinx没有的优点——不需要额外建立索引的过程。如果是初学搜索，Xapian是个不错的选择。

Sphinx：就代码风格上，Sphinx几乎是所有这些提到的项目里面最差的。在早期版本中，构建索引就是一个几千行的大函数。可Sphinx几乎完全符合我的选型标准，而且是当时我能找到的开源世界中唯一符合标准的项目。代码的优雅让位于现实工程上的考虑，因此Coreseek现在的原系统是基于Sphinx的。此外，为了克服Sphinx支持的数据来源单一的问题，我们又在上面开发了Python的数据源接口。需要指出的是，Sphinx并非我们第一个介绍进入国内的，只是在我们之前，每段介绍的文字都会说类似的话，“很好，很好，只是不支持中文。”

选择Sphinx还有另一个因素，在Coresek最初的时间里，基于全文检索技术创业的公司非常多，绝大多数公司（在我所知是100%）都是基于Lucene进行了二次开发，我非常不希望被人看成 Yet Another Lucene Company（看啊，又一个Lucene公司）。

修改Sphinx加入中文分词，并不难，一共只用了不到一周。现在，我们有了分词法和全文引擎，该是拉出来溜溜的时候了。

商业化历程

2008年3月，Coreseek网站上线。用这个事件作为起点有点儿小讽刺，因为网站上线，开源的原因，并非要开始进行商业化。而是——“在全文检索上，我已经投入了这么多时间，我却还不知道怎么能从上面挣到钱。于是开源出来，如果有人觉得有用，就拿去用吧。”应该说，事实上，开源的时候也是我几乎已打算放弃的时候。但不管怎样，总要试一下才知道吧。

开源之后，找到用户来用，成为了摆在我面前最大的问题。我尝试过各种方法去让别人知道Coreseek/Sphinx。

给站长发邮件：可能有些站长的邮箱里，还有这封信。大意是我们发现您的网站在全文检索功能上还有可以改进的空间，我们这里有一个开源免费的检索系统，希望能够帮到您。如果您不清楚怎么用，我们可以帮您安装，而且是免费的。效果是：没有回应。

> 我尝试了很多种方式推广项目，比如给站长发邮件、参加会议、技术沙龙等，均没有得到明显的商业进展。我像孤单的地球在发射电波“你们在哪儿？我在这儿……”

发帖子：内容和邮件的内容类似。效果是：几个回帖。“你那个东西真的能用吗？”

扫楼：卖保险的、推销信用卡的，通常会采用扫楼的方式展开业务。而用扫楼的方式推销开源软件，也许我是头一个。印了不少推广的文字材料，发给各家IT公司的前台。效果是：没有回应。

参加会议：由于前面的种种不顺，我分析是接触的人员在组织结构中的层级太低，而检索系统选型，基本上是需要技术经理或CTO决策的事情。为此，我去参加各种IT行业的聚会，借着交换名片的机会，推销Coreseek/Sphinx，因为在换名片的过程中，总要介绍自己是做什么的吧？这种方法的效果不错，认识了不少好朋友，Coreseek也多了一些用户。可惜，在商业上的进展，仍然是零。

技术沙龙：与比其他方式相比，这种方式效率最高。因为参加技术聚会的人一般有一定的目的性，是带着问题而来的。只要你的产品能够切实解决他们的问题，就能争取到不少新用户。这种方式的问题在于，参会的往往是些技术发烧友，离主流商业用户还有一段距离。

在推广项目时，感觉自己就像是孤单的地球，漫无目的地向宇宙中发射电波：“你们在哪儿？我在这儿……”

这么不温不火地过了几个月，直到突然有一天晚

上，我接到了一个电话。“你的那个检索系统稳定吗？这里是ChinaUnix，我们想用。”这是第一个收到钱的用户，也是第一个有较大流量的网站用户。

有了这个典型用户，后面的事情相对容易一些了。在被一些大网站采用后，国内主要的PHP程序也都默认提供了Sphinx的检索接口。

其中，需要特别感谢的是时任Blogbus CTO的车东，项目在Blogbus实施的示范作用，以及他在博客上对Sphinx的介绍为项目推广过程带来了不可估量的帮助；和时任ITEye CEO的范凯，在他组织的国产开源软件介绍专题中给了我们一个专题，这个专题放在ITEye网站首页相当长的一段时间。

易用性往往不影响用户最终的选择决策，而且与大型商业公司相比，开源软件厂商的资源通常非常有限，因此不要在易用性上浪费宝贵的资源，将资源用在那些必须做的功能上。

在被大型网站采用后，收入开始逐渐稳定了起来。虽然远远不是一个成功的公司，但作为一个开源项目来说，至少足以糊口，维持发展。

开源系统看起来很美

后来，在技术人员的聚会上，时常有人和我打招呼：“我们在某某系统用了你们的东西，很不错。项目进行得很顺利。”我当然报以微笑，但心里不禁会想：“能对你们有用，我很高兴；如果你们愿意付点钱，我就更高兴了。”这在某种程度上，也隐含地暗示出了开源软件作为一个生态圈存在的问题。在“开源传万世，因有我参与”的激扬与高亢之后，脚总要落在地面上。

下面是从我个人的经验出发，认为开源系统需要特别注意的几个问题。

易用性陷阱

我曾经和国内某大型PHP软件厂商接触，他们第一个问题是：“你们的系统易用性如何？好不好安装？”安装、配置的易用性对于IT厂商的安装人员很重要，容易安装也意味着实施人员成本的降低。但对于开源软件厂商来说，易用性却可能是一个陷阱。

其一，容易使用必然带来大量的界面、配置、容错等方面的开发工作，这导致开源软件的厂商或作者不得不把时间投入在可能只使用一次的代码模块上。

其二，易用性往往不影响用户最终的选择决策。直到现在，Apache、MySQL、Nginx等软件，仍然通过命令行和配置文件，而非图形界面进行安装和配置。用户选择这些开源软件的理由，并非如何易用，而是这些软件能够解决他们工作中的问题。而由于开源软件厂商不直接和客户打交道，所以往往会被中间的集成商“忽悠”，在易用性上浪费资源。

开源软件厂商不比大型商业公司，资源通常非常有限，不要在易用性上浪费宝贵的资源，将资源用在那些必须做的功能上。

类似的易用性陷阱还很多，例如：可配置、插件化、可定制。以MySQL为例，其插件化的体系结构是它成功或者说吸引第三方开发者使用的一个因素，但现在来看，因为其插件化的体系结构，导致缺少全局的优化方式，最终导致整个体系在底层持续膨胀而非收敛的。而这些看似强大的功能会带来额外的设计和调试的复杂度，反而影响实际问题的解决。

如果把最终用户使用软件的过程看做一个最优化的过程，那么我们需要提供的不是一个个选项，“你可以这样，也可以那样”，而是一个在用户特定问题领域下的最优解。

开源与商业模式

开源软件公司往往是技术人员合伙，这个架构天生就有一个缺陷：缺少市场人员，导难以面向最终用户进行销售。并且由于人员短缺，也往往无力承担整个项目，只能作为一个大项目的一部分。又因开源，在市场上并没有太强的议价权。这种种因素，导致了开源软件企业很难拿到项目

中利润最丰厚的部分。以MySQL为例，其用户到客户的转化率不到1%（传言）；而我们的数据就更难看了，同样的用户基数，我们的营业额不足对手公司的1/10。

传统观点认为，开源软件可以通过服务、咨询等多种方式收费，而从我们自己的实践来看，如果仅仅是作为提供技术支持型的公司（可以类比计算机硬件的维修铺子），这种商业模型是可行的。但如果是自己研发，特别是需要研发开源世界之前从未有过的功能，这种商业模型提供的利润不足以支持高昂的研发成本。以Coreseek为例，为了研发下一代检索系统，仅需求调研、原型系统分析，就已投入了10多个人月。

国外一些开源软件公司改为了部分开源的模式，对于一些基本功能开源，对于企业应用必须的模块和特性，就必须要购买许可。这也许是解决开源公司缺少市场议价权的路子。

开源软件隐含的限制

因为软件开源，所以进入开源服务这个市场，基本不会有什么门槛，这导致了从事服务的人员素质参差不齐，甚至有些人连用户手册都没读全就开始给用户服务了。这给整个行业造成了非常不利的影响。正如我们对事物的了解不会一蹴而就一样，从事开源软件服务或咨询，不但要了解这个系统的优点，更要了解系统的缺点和限制，才能够有的放矢。

例如，Google的LevelDB，开源之后，应者云集。非常多的数据库服务把后端的存储引擎改到了LevelDB上，可LevelDB在处理文件句柄上存在漏洞，而这个漏洞是本来不应该出现在类数据库软件中的。

类似的，在Lucene、Sphinx、MySQL、Postgresql、Hypertable等系统中，都有这样或那样的设计限制，可这些限制往往并没有形成文档，只能通过阅读代码发现和了解。

使用没有审查过设计、没有大体通读过源码的开源软件，无异于在项目中埋了枚地雷。而开源软件公司的服务价值，并非仅仅是让系统能够运行，而是发现开源系统中隐含的限制是否影响当前项目的使用，以及出现了问题，该如何修复。

靠开源确实可以吸引注意力，也可以吸引一些早期尝鲜性质的用户，但把一个开源系统变成一个有持续生命力的系统，成为一个真正在用户处发挥实际价值的系统，需要不亚于一个商业软件推广力量上的投入。而开源，仅仅是这一系列工作的第一步。

开源并不意味着别人会来主动帮助你，开源是一种姿态，是一种尚处于弱小阶段的初创公司的一种姿态——“有我在，你就在；我不在，你还在。”

展望

经历了4年多的开源摸索，Coreseek在开发、实施Coreseek/Sphinx系统的过程中遇到了一些问题。根据客户的反馈意见，我们在设计下一代的检索产品。Coreseek的检索系统将不再是一个外挂的全文检索模块，而可以成为整个应用系统运行的平台。具体特性有：索引的自动分区、分布式系统的全局统计视图、支持PB级的内容检索请求等。

致谢

感谢这些年里给予我们无私帮助的那些可敬的人们。在赵海博士的帮助下，我了解了条件随机场以及他提出的6字窗口6标注集分词系统；是车东与范凯的帮助，才让Coreseek/Sphinx项目被国内很多开发者所知。

最该感谢的是你们，我们可爱的客户和捐助者们，是你们的选择和信任，才让我们可以一直走到今天。

李沫南

系统分析师，Coreseek创始人。主要从事浏览器、分布式检索系统、中文自然语言处理等方面的研发工作。

Muduo：多核时代的C++网络编程

文 / 陈硕

Muduo是一个个人开源项目，它是一个现代的非阻塞事件驱动C++网络库，原生支持多核多线程，运行在32-bit/64-bit Linux上。以x86-64为主要目标平台，兼顾32-bit ARM和x86。

接口、功能、性能

现代C++接口

Muduo采用了现代C++接口，主要体现在两方面：事件回调与对象生命期管理。

在传统的C++程序中，事件回调是通过虚函数进行的。网络库往往会定义一个或几个抽象基类（Handler class），其中声明了一些（纯）虚函数，如onConnect()、onDisconnect()、onMessage()、onTimer()等。使用者需要继承这些基类，并覆写（override）这些虚函数，以获得事件回调通知。由于C++的动态绑定只能通过指针和引用实现，使用者必须把派生类（MyHandler）对象的指针或引用隐式转换为基类（Handler）的指针或引用，再注册到网络库中。MyHandler对象通常是动态创建的，位于堆上，用完后需要delete。网络库调用基类的虚函数，通过动态绑定机制实际调用的是用户在派生类中覆写的虚函数，这也是各种OO Framework的通行做法。这种方式在Java这种纯面向对象语言中是正当做法（Java 8也有新的Closure语法，C#从一开始就有delegate）。但在C++这种非GC语言中，使用虚函数作为事件回调接口有其本质困难，即派生类对象的生命期管理。在这种接口风格中，MyHandler对象的所有权和生命期很模糊，到底谁（用户还是网络库）有权利释放它呢？有的网络库甚至有delete this；这种做法，让人捏一把汗：如何才能保证此刻程序其他地方处没有保存这个即将销毁的对象的指针呢？另外如果网络库需要自己创建MyHandler对象（比方说需要为每个TCP连接创建一个MyHandler对象），那么就得定义另一个抽象基类HandlerFactory，用户要从它派生出MyHandlerFactory，再把后者的指针或引用注册到网络库中。以上这些都是面向对象编程的常规思路，或许大家早已烂熟于心。

在现代C++中（指2005年TR1之后，不是最新的C++11），事件回调有了新的推荐做法——boost::function+boost::bind（即std::tr1::function+std::tr1::bind，也是最新C++11中的std::function+std::bind），这种方式的优点见我写的《以boost::function和boost:bind 取代虚函数》和孟岩的《function/bind的救赎（上）》。Muduo正是用boost::function来表示事件回调，包括TCP网络编程的三个半I/O事件和定时器事件等。用户代码可以传入签名相同的全局函数，也可以借助boost::bind把对象的成员函数传给网络库作为事件回调的接受方。这种接口方式对用户代码的class类型没有限制（不必从特定的基类派生），对成员函数名也没有限制，只对函数签名有部分限制。这样自然也解决了空悬指针的难题，因为传给网络库的都是具有值语义的boost::function对象。从这个意义上说，Muduo不是一个面向对象的库，而是一个基于对象的库。因为Muduo暴露的接口都是一个个的具体类，完全没有虚函数（无论是调用还是回调）。

现代C++的另一个特点是对象生命期管理的进步，体现在不需要手工delete对象。在网络编程中，有的对象是长命的（例如TcpServer），有的对象是短命的（例如TcpConnection）。长命对象

的生命期往往和整个程序一样长，很容易处理，直接使用全局对象（或scoped_ptr）或者做成main()的栈上对象都行。对于短命对象，其生命期不一定完全由我们控制，比如对方客户端断开了某个TCP socket，它对应的服务端进程中的TcpConnection对象的生命也即将走到尽头。但这时我们并不能立刻delete这个对象，因为其他地方可能还持有它的引用，冒然delete会造成空悬指针。只有确保其他地方没有持有该对象的引用，才能安全地销毁对象，这自然会用到引用计数。在多线程程序中，安全地销毁对象不是一件轻而易举的事情，见我写的博文《当析构函数遇到多线程——C++中线程安全的对象回调》。

在非阻塞网络编程中，我们常常要面临这样一种场景：从某个TCP连接A收到了一个request，程序开始处理这个request；处理可能要花一定的时间，为了避免耽误处理其他request，程序记住了发来request的TCP连接，在某个线程池中处理这个请求；在处理完之后，会把response发回TCP连接A。但在处理request的过程中，客户端断开了TCP连接A，而另一个客户端刚好创建了新连接B。我们的程序不能只记住TCP连接A的文件描述符，而应该持有封装socket连接的TcpConnection对象，保证在处理request期间TCP连接A的文件描述符不会被关闭。

否则的话，旧的TCP连接A一断开，TcpConnection对象销毁，关闭了旧的文件描述符（RAII），而且新连接B的socket文件描述符有可能等于之前断开的TCP连接（这是完全可能的，POSIX要求每次新建文件描述符时选取当前最小的可用的整数）。当程序处理完旧连接的request时，就有可能把response发给新的TCP连接B，造成串话。

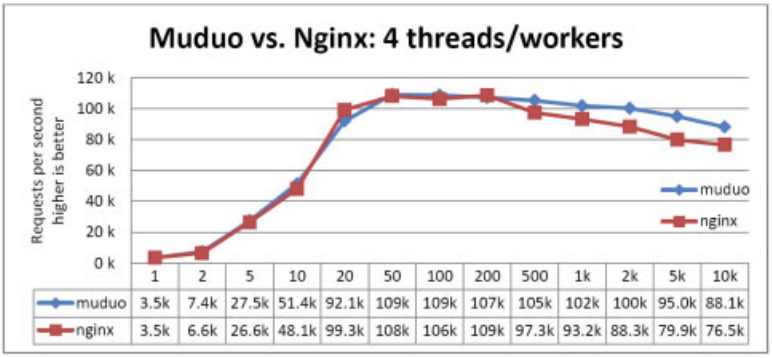
为了应对这种情况，防止访问失效的对象或者发生网络串话，Muduo使用shared_ptr来管理TcpConnection的生命期。这是唯一一个采用引用计数方式管理生命期的对象。如果不用shared_ptr，我想不出其他安全且高效的办法来管理多线程网络服务端程序中的并发连接。

功能

Muduo不是那种大而全的网络库，它不跨平台，不支持UDP（我认为UDP的用途有两个，一是传输实时多媒体数据，容忍数据丢失；二是NAT穿透。这二者都不是Muduo的目标应用领域，而且UDP与TCP有着完全不同的多线程并发模型，很难兼顾）和串口，也不支持IPv6。Muduo有着明确的适用范围：编写企业内部的基于Linux的分布式系统，例如实现分布式infrastructure或服务端网络应用程序。Muduo特别适合处理大量TCP长连接，没有为TCP短连接优化。Muduo只有一种I/O方式：配合I/O multiplexing的非阻塞I/O，它采用one event loop per thread作为基本的多线程并发模型。Muduo原生支持多核多线程，见“线程模型”部分的详细介绍。

性能

好于其他Native C/C++通用网络库（libevent2、Boost.Asio）。在某些情况下可与以性能著称的Nginx、ZeroMQ一较长短（如图1所示）。



	1	2	5	10	20	50	100	200	500	1k	2k	5k	10k
muduo	3.5k	7.4k	27.5k	51.4k	92.1k	109k	109k	107k	105k	102k	100k	95.0k	88.1k
nginx	3.5k	6.6k	26.6k	48.1k	99.3k	108k	106k	109k	97.3k	93.2k	88.3k	79.9k	76.5k

图1 Muduo与Nginx多线程/多进程性能对比

参加对比的是Nginx 1.0.12，采用http echo模块，直接在内存中响应请求，不访问硬盘。

线程模型

现在服务端网络编程处理并发连接主要有两种方式。

- 当“线程”很廉价时，一台机器上可以创建远高于CPU数目的“线程”【注：不是操作系统的原生线程，不能被操作系统的任务调度器看见，通常由语言的runtime自行调度】。这时一个线程只处理一个TCP连接（甚至半个），通常使用阻塞I/O。例如Python gevent、Goroutine、Erlang actor。
- 当线程很宝贵时，一台机器上只能创建与CPU数目相当的线程。这时一个线程要处理多个TCP连接

上的I/O，通常使用非阻塞I/O和I/O multiplexing。例如Java Netty和libevent。这是原生线程，能被操作系统的任务调度器看见。

注意，上面列出的网络编程方案不是每个都能自动发挥多核的性能优势。

Muduo采用的是第二种方式，这是在Linux下使用Native语言编写用户态高性能网络程序的最成熟的模式。在单线程时代，这种模式称为Reactor（由ACE的作者Douglas Schmidt提出的，图2）；在多线程时代，libev的作者Marc Lehmann把它扩展为one event loop per thread（图4）。Muduo采用的正是这种思路，并且I/O线程可以与线程池配合（图3和图5）。

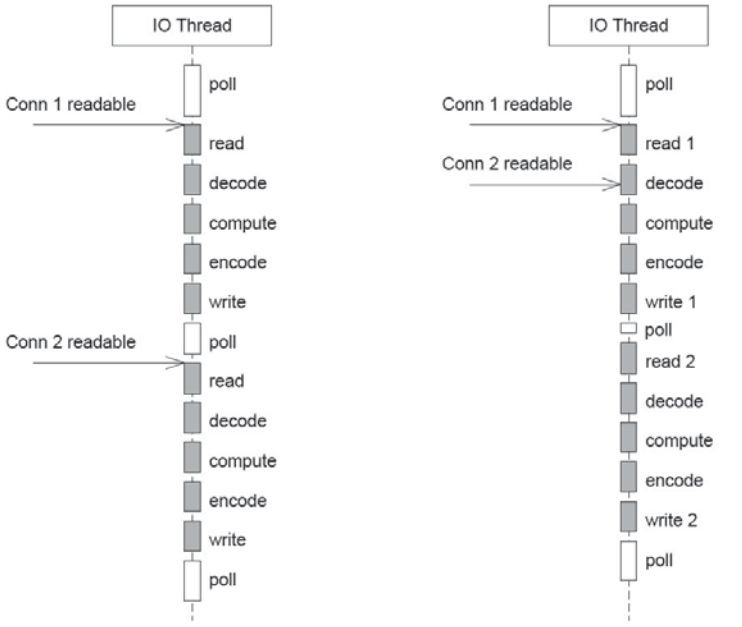


图2 传统单线程非阻塞网络编程的事件处理模型

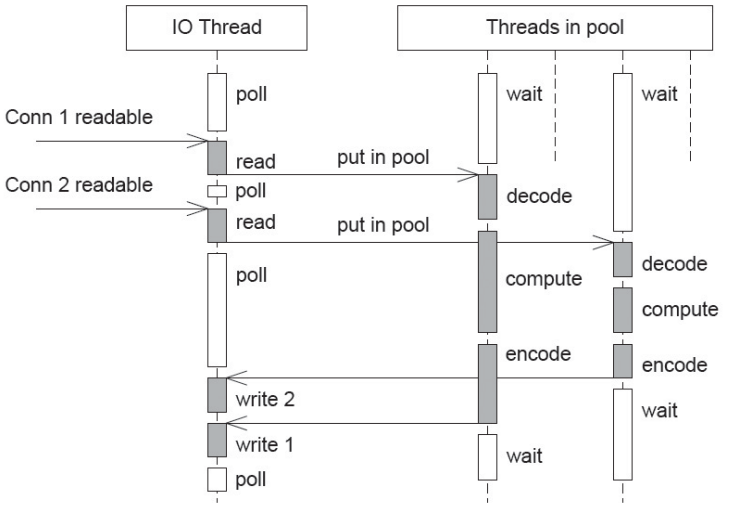


图3 单线程I／O与线程池配合

限于篇幅，这里不详细讨论这几种方案，而是用银行柜台办理业务为比喻，简述各种模型的特点。银行有旋转门，办理业务的客户人员从旋转门进出（I/O）；银行有柜台，客户在柜台办理业务（计算）。要想办理业务，客户要先通过旋转门进入银行；办理完之后，客户要再次通过旋转门离开银行。一个客户可以办理多次业务，每次都必须从旋转门进出（TCP长连接）。另外，旋转门一次只允许一个客户通过（无论进出），因为read/write只能同时调用其中一个。

- 这家小银行有一个旋转门，一个柜台，每次只允许一名客户办理业务。而且当有人在办理业务时，旋转门是锁住的（计算和I/O在同一线程）。为了维持工作效率，银行要求客户尽快办理业务，最好不要在取款时打电话问家里人密码，也不要在通过旋转门时停下来系鞋带，这都会阻塞其他堵在门外的客户。如果客户很少，这是很经济且高效的方案；但如果场地较大（多核），那么这种布局就浪费了不少资源，只能并发不能并行。如果确实一次办不完，应该离开柜台，到门外等着，等银行通知再来继续办理（分阶段回调）。

- 这家银行有一个旋转门，一个或多个柜台。银行进门之后有一个队列，客户在这里排队到柜台（线程池）办理业务。即在单线程Reactor后面有一个线程池用于计算，可以利用多核。旋转门基本是不锁的，随时都可以进出。但排队会消耗一点时间，相比之下，第1种情况客户一进门就能立刻办理业务。

- 这家大银行相当于包含图2中的多家小银行，每个客户进大门时就被固定分配到某一家小银行中，他的业务只能由这家小银行办理，他每次都要进出小银行的旋转门。但总体来看，大银行可以同时服务多个客户。这种同样要求办理业务时不能空等（阻塞），否则会影响分到同一家小银行的其他客户。而且必要时可以为VIP客户单独开一家或几家小银行，优先办理VIP业务。这跟第1种情况不同，当普通客户在办理业务的时候，VIP客户也只

能在门外等着（图2右）。这是一种适应性很强的方案，也是Muduo原生的多线程I/O模型。

■ 这家大银行有多个旋转门，多个柜台。旋转门和柜台之间没有一一对应关系，客户进大门时就被固定分配到某一旋转门中（奇怪的安排），进入旋转门之后，有一个队列，客户在此排队到柜台办理业务。这种方案的资源利用率可能比第3种方案更高，一个客户不会被同一小银行的其他客户阻塞，但延迟也比第3种方案大。

以上粗略介绍了Muduo的默认多线程模型，详细的分析见Muduo手册中《详解Muduo多线程模型》一节。

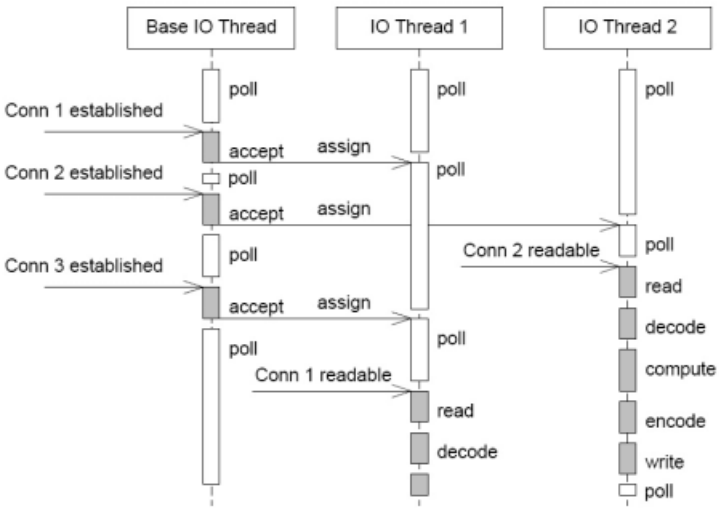


图4 Multi-loop

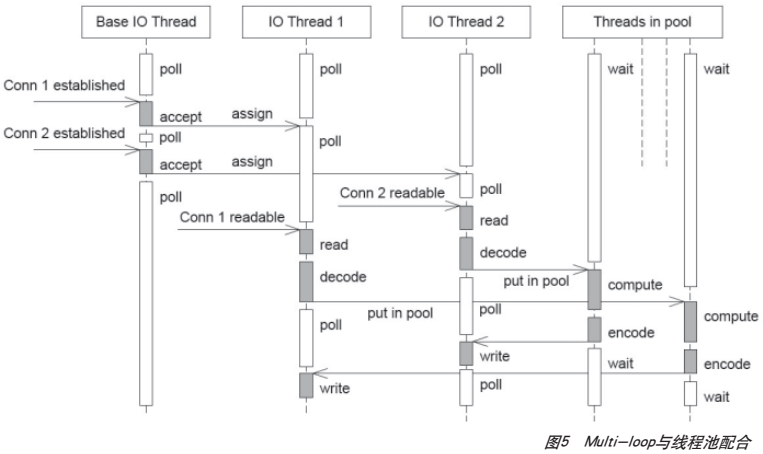


图5 Multi-loop与线程池配合

特点总结

■ 32-bit/64-bit兼顾。以64-bit的Intel x86为首要目标平台，这是目前开发服务端网络应用的主流平台，同时兼顾32-bit ARM平台。

■ 使用了Linux平台专有的epoll、eventfd、timerfd等系统调用，性能较好。

■ 代码易读。Muduo网络库的核心代码只有不到5000行，没有层峦叠嶂的class hierarchy，也没有使用异常处理。代码里都是一个个具体类，易于分析。代码在最高警告级别下干净地编译通过。

■ 多核多线程友好。一开始设计就考虑支持多线程，在关键路径上无锁。

■ 示例丰富。有各种范例近20个，编译出近百个可执行文件，涵盖常见的TCP网络编程任务，包括用事件的方式处理signal等。

■ 具备扩展性。借助编解码器（codec）可支持Google Protobuf等二进制格式。Muduo的I/O事件循环可融合其他非阻塞事件驱动的现有网络库，例如异步DNS、curl、数据库客户端等。

扩展项目：http://github.com/chenshuo/muduo-protorpc实现Google protobuf RPC。

更多信息

项目主页：http://code.google.com/p/muduo

140页文档：http://126.am/muduo

30分钟演讲：http://blog.csdn.net/solstice/article/details/7703959

陈硕

现任职于香港某跨国金融公司IT部门，从事实时外汇交易系统开发。编写了开源C++网络库Muduo；参与翻译了《代码大全（第二版）》和《C++编程规范（繁体版）》，整理了《C++ Primer第4版评注版》。

应对大数据，数据库在行动

来自对IBM信息管理团队的采访

记者 / 董世晓

数据库，尤其是关系型数据库，经过40余年的发展，产品技术已经比较成熟和完善，在保持数据的一致性（事务处理）、数据更新开销、JOIN等复杂查询方面具有明显的优势，在各行业得到了广泛的应用并得到充分的认可。从2000年以来，伴随着互联网行业的快速发展，带来了“大数据”的难题：现在每天都会产生超过数十PB的新信息，这其中80%以上都是各种新的纷繁芜杂的非关系型数据，例如电子邮件、文档、多媒体、RFID源等，而关系型数据库在处理这些非关系型数据时，已显现出力不从心。如何更好地处理大数据，摆到了每家数据库软件厂商、数据库用户的案头。从用户的需求来说，迫切希望数据库软件能够处理这些更多、更复杂类型的数据，以实现更好的数据管理。那么，数据库软件未来的方向在哪里？关系型数据库如何应对大数据以做出变革？大数据时代，数据库该怎样选型？带着这几个问题，我们采访了来自信息管理工作一线的IBM数据管理团队资深信息顾问王敏和大数据技术顾问陈威。

IBM数据管理团队大数据技术顾问陈威：Hadoop只是大数据的小模块

数据库软件的方向

在王敏看来，正是由于大数据的出现，对关系型数据库提出了更多的需求，并因此成为推动数据库技术不断发展的主要动力。这体现在两个方面：一方面，数据库软件在持续地增加对新技术的支持，包括从早期的对二进制数据的支持、对音视频的支持、XML技术的推出，到自动管理、自动压缩等能力，其实都是为了适应新技术而做出的技术支持，将新的技术容纳进来，尽量让数据库变得更强大；另一方面，数据库在不断出现分支，不同领域或者不同方向的产品能够更好地满足某些特定的需求，就像树一样最后会长出不同的分支，而这也是为了适应专业化的需求。事实上，数据管理领域也会切分出不同的方向去处理不同的问题，例如面向交易、面向分析或者面向快速性能等的数据库，以及运行更小型设备（例如单片机、手机）上的微型数据库、移动数据库，甚至软硬件一体化的数据库例如Netezza，所有这些都代表了数据库专业化分工的发展方向。

对此，陈威补充道，在谈数据库时，要做到“Fit for Purpose”，不能像以前那样坚持“大一统”，要针对不同的需求和领域，进行更细致、更专业化的分工，去满足客户的需求。

SQL、NoSQL和NewSQL

谈到关系型数据库（这里用SQL描述），就不得不提NoSQL以及近来谈论较多的NewSQL。

陈威认为，SQL在处理核心业务方面，具有非常大的优势，而NoSQL会更强调低成本的扩展性。但目前看来，一方面，SQL产品正在更多融

入NoSQL技术，例如Hadoop等；另一方面，很多NoSQL产品所做的一些更新，也正在向SQL靠近。这两方面的力量，最终将会形成合力，催生NewSQL。目前，IBM的一个重要研究方向就是NewSQL。说到底，数据库是为了存数据，只不过针对不同的数据格式会有不同的优势，各种技术相互借鉴和学习，并最终融合到主流技术中去。因此SQL、NoSQL、NewSQL之间并不冲突，只是处于不同的数据管理领域或者层次。

关于这个问题，王敏提出了八字诀——“相互补充，共同发展”。有些新兴技术可能发展得快，但同时可能消亡得也快，例如曾经很火的面向对象的数据库，现在基本上很势微。新技术确实在一定时间内迎合了一些需求，但它能否可持续发展并保持生命力，取决于多种因素，不仅是技术，还有很多商业因素。但综合来看，历史总是螺旋上升的。

Hadoop只是大数据的一个小模块

既然NewSQL是IBM未来的一个发展方向，而它又是SQL和NoSQL相互融合而形成的结果，那么IBM在其传统关系型数据库DB2基础上，吸收了哪些NoSQL的元素呢？

事实上，与其他关系型数据库厂商相比，IBM在业界率先推出了基于NoSQL的商业版本——InfoSphere BigInsights。BigInsights基于开源的Hadoop，HBase、Hive也应用其中。在知名第三方最新的研究报告中，IBM处于Hadoop解决方案的领导者象限。而针对HBase在Column Family太多时性能不佳这个问题，IBM还特别研发了一项新技术，以作为对HBase的一个扩充。

当然，陈威也提到，Hadoop只是大数据处理的一个方向、一个小模块而已，不代表整个大数据的方案。因此，IBM的大数据平台并不仅仅局限于Hadoop，还涵盖诸多其他内容，这其中最为有代表性的就是流计算（Stream Computing）。众所周知，大数据具有Volume、Variety、Velocity三个维度，而针对Velocity这个维度，IBM最拿手的就是流计算。流计算可用于处理流动中的、不落地的数据，包括像电信CDR记录、天文数据、GPS信息、证交所数据等量级巨大且不断变化的数据，这些数

IBM数据管理团队资深信息顾问王敏：数据库选型原则不变

据已没有办法用传统的方式存储完以后再做分析，因为那会相当耗时，可能要T+1甚至T+2才能完成。显然，在其对应的场景中，是不允许出现这样的结果的。而据陈威披露，流计算在IBM已有七八年的历史，在国外拥有众多客户，而在国内正处于蓬勃发展期。

数据库选型原则不变

在大数据时代，打着大数据处理旗号的数据库产品琳琅满目，迷了人眼，究竟该怎样做数据库选型呢？

在王敏看来，数据库产品虽多，但用户在进行数据库产品选型时，不要乱了方寸，应该依然坚守五个**原则**：成熟度、先进性、可靠性、易用性和维护成本。

成熟度：要保证产品是经过长期实践验证的。

先进性：产品要充满活力，以能够持续支撑新需求和新应用。事实上如果产品不够先进，就会使得用户为了满足新需求而不得不自己去做一些额外的开发工作。

可靠性：安全可靠，不轻易宕机。对数据库，尤其是交易型的数据库来说，可靠性是非常关键的指标。

易用性：涉及管理、开发等方面，是考核一个产品好不好用的重要标准。

维护成本：要看维护管理是否方便，要具备更多的自动化管理功能，减少机械性的工作，减轻DBA的管理负担。

人才招聘新趋势：垂直性的社交网络

pongo网（庞果网）CEO李炯明专访

记者 / 长卿

近日，国内专业IT人才招聘服务公司Careerfocus联合全球最大中文IT社区CSDN推出了IT行业细分的招聘求职网站pongo（庞果网），力图通过搜索引擎，垂直型社交网络构建IT行业企业与个人专属性的招聘与求职社区。《程序员》记者专访了pongo网CEO李炯明，请他就当前IT企业招聘中面临的挑战以及在线人才招聘发展趋势分享了自己的观点。

人才招聘最大的问题与障碍：缺乏IT行业特征的垂直招聘社区

《程序员》：人才招聘是当前大部分企业头疼的难题，在您看来，IT企业招聘中的难点是什么？

李炯明：中国互联网发展了10多年，但是没有凸显“行业特性”价值的、行业垂直化程度高的IT招聘求职社区。综合性招聘门户的有效性在下降，其原因一方面是综合性招聘网站不能提供行业特有的垂直价值；另一方面，互联网用户在不断地向社交网络转移，传统的Web1.0网站带给用户的服务太过于单一。而80%的中小IT企业对传统招聘网站是过度依赖的，这就是为什么大多数IT企业长期面临招人困难的重要原因。

不同层次的人员招聘难易程度不同。大部分初级技术性人才是主动求职者，但诸如CTO、CIO、技术总监、高级架构师以及优秀的产品经理这样的中高级人才往往是被动的求职者，要通过传统的招聘网站找到这些职位的人才就太难了。

《程序员》：垂直型的招聘网站在其他国家的发展状况是怎样的？在中国为什么还未出现大量细分、垂直领域的人才招聘求职网站？

李炯明：在美国与日本，垂直型的招聘网站已非常细分，有针对女性招聘的、有满足临时性工作的，也有面向各个行业的……

一个招聘门户要成功，最重要的是能获取足够多的个人注册用户。传统招聘门户的投入主要是在市场营销上，它获取注册用户的方式是通过广告来驱动：无论线上、线下、户外、电梯、汽车车身……到处都是广告，这会导致其个人注册用户的获取成本极高。而在目标人群较为细分的领域，如果采用同样的方式，会造成营销资源的大量浪费收效却甚微。

这就是垂直型人才网站发展的难度和门槛：它的市场费用高昂，而获取人才和注册用户数的效率相对较低。但一旦垂直型网站经营成功之后，它带给雇主和求职者的价值远大于通用型网站。

《程序员》：垂直性社区到底能为招聘求职提供什么样的服务价值？

李炯明：很显然，无论51job、智联招聘这些网站如何更改其“搜索”算法或结果呈现方式，得到的信息一定只能是单一的。但pongo希望能够呈现的内容包括职位、圈子、人、博客、文章、问答、课程等方面，这些要素可以通过“搜索”行为获得或者通过社区个人用户关系链来传播，这是pongo未来要做的事情以及使命，从这个意义上讲，pongo会为每一个用户建立“职业轴线”。所有知识以及知识分享都是围绕个体的职业技能、职业能力需要。这些结果的呈现，必须依赖一个垂直的、相对成熟的UGC社区。很显然，在中国IT社区中只有CSDN具备这样的雏形，以及有可能转化成为IT招聘求职垂直型社区。

未来在线招聘与求职服务，为什么一定会是社交网络模式？

《程序员》：传统Web1.0招聘网站有什么局限性？社交网络对招聘求职有怎样的影响与冲击？

李炯明：中国知名的IT企业，例如腾讯、百度、阿里巴巴、盛大等，每年社会招聘中有30%~45%来自“内部员工推荐”；而根据我们对2000名专业人士的调查，46%的被调查对象说他们“最近这份工作”是通过朋友或者相关人士引荐获得的。这两个数据再一次说明了“弱关系”在招聘与求职过程的重要价值，从“现实逻辑”上肯定了社交网络对于招聘与求职的应用价值。

传统招聘网站的最大问题在于用户关系得不到强化。刚刚谈到，企业招聘很有效的渠道之一是“推荐”，而推荐行为背后需要特定的关系作支撑。所以在未来，求职平台能否良好运行与如何构建和呈现这些关系属性有很大的联系，而传统的Web1.0架构无法实现这一切，必须依靠新型的社区化平台。

同时，作为传统的招聘网站，必须重新思考一个问题：我们能为用户带来什么价值？实际上，我们有必要为用户提供多样化的价值和服务。

长远来看，这种社交化的平台将成为整个招聘求职的最主要形态。

专注IT企业与信息技术领域的专业人士，做垂直化服务

《程序员》：你为什么选择在这个时机切入IT行业的人才招聘与求职领域？

李炯明：首先，我认为中国市场对专业性强IT招聘求职细分网站有较高的刚性需求，IT企业的招聘效率普遍比较低，大家都希望有这样一家网站来提高企业的招聘效率；其次，十几年来，在线招聘服务一直由通用型招聘网站控制着竞争和服务的格局，所以垂直型招聘服务一定有很大的发展空间和机会；再次，IT行业人才的招聘需求一直非常旺盛，占到了整个招聘求职市场份额的30%~35%，所以其市场容量非常大。

《程序员》：您创办的Pongo是一个什么样的网站？它与当前综合型招聘门户51job、智联招聘等网站有什么不同？

李炯明：“pongo”意为“猿猴”，因为程序员喜欢把自己戏称为“程序猿”，所以我们选择了“pongo”作为网站名。这个项目从2012年4月开始，在7月8日上线。它是完全基于CSDN用户、垂直于IT领域的招聘求职网站，和CSDN拥有统一的账号系统和个人用户数据。

pongo只为IT行业的个人求职者和有IT类职位需求的雇主提供服务。同时我们的服务形式也与传统招聘网站不同，对个人求职者而言，我们不仅能提供职位搜索服务，还能获取资讯、分享知识、参与问答、参加编程游戏，尤其还包含社会化的推荐、悬赏等元素。同时，我们与CSDN的其他服务无缝融合，如科技资讯、社区、下载、空间、项目外包等。也正因此，普通传统招聘网站无论在用户数据量、垂直行业的影响力以及垂直领域服务的多样性方面，都无法与我们匹敌。基于CSDN社区多年的行业积累，我们专注于IT企业和专业开发者，做好垂直化的人才服务。

《程序员》：在您看来，如何在国内做好垂直型人才招聘网站？

李炯明：作为垂直性人才网站的创始和运营团队，我认为首先要考虑清楚以下四个问题。

第一，垂直型招聘网站跟传统招聘网站相比，其获取个人用户的渠道和方式应该不一样，不能完全依赖于广告驱动。

第二，业务必须建立在现有的垂直型社区化服务的基础上，这样成功的可能性更大。

第三，提供给个人用户的价值一定是多元化的。因为每个人的求职周期是2~3年，甚至更长。从这个意义上来看，一个垂直型招聘网站在底层应按照社交网络的架构来搭建，否则无法把多样化的服务和内容传递给个人用户。

第四，平台要考虑一定程度的开放。提供开放接口给第三方的服务，例如科技类培训、项目管理培训、产品培训等。

分布式敏捷团队的经验分享

文 / 胡凯

本文分享了ThoughtWorks中国一支50人规模的团队在分布式环境下进行敏捷实践所遇到的问题，同时给出了该团队摸索出的解决方案。

realestate.com.au（简称REA）是澳洲最大的房地产垂直搜索和广告平台，它与ThoughtWorks西安在一年半以前开始合作。尽管REA同样是敏捷成熟度非常高的组织，但ThoughtWorks与REA的合作也历经一年多的摸索和调整才渐渐走上正轨。回头去看这一年中经历的困难和挑战，很大一部分是分布式环境带来的，而团队规模则扮演了困难放大器的角色。

究竟哪些敏捷实践在分布式环境中比较脆弱、容易走样呢？该如何解决？让我们用极限编程的洋葱圈模型来探讨这个问题。从内到外，洋葱圈模型将敏捷实践划分成了个人、团队和组织三个层面。而据我的观察，分布式环境通常会对组织和团队层面的敏捷实践产生直接或间接影响。

无间断视频连接起来的完整团队

完整团队意味着在组织团队时尽量将所需的各种角色配置到团队中，起到减少交流壁垒、缩短反馈周期、增加决策及时性与合理性的效果。我们团队的现状是界面设计师、开发工程师、测试工程师和一部分业务分析师在西安，而产品负责人、客户代表等在澳洲。分布式开发环境造成了中澳双方都无法建立完整团队，大部分工作需要双方配合完成。随之而来的问题是：

- E-mail常常无人响应；
- 问题需要来回多次才能得到确认，而时差和假日则常常把这段时间拉长为数日；
- 多方参与的活动在时差的影响下难以计划和实施，比如验收测试、优先级调整和功能验收等。

就我的观察来看，这是急事与要事博弈的结果。理论上要事应当优先，但事情的紧急性更容易感知，产生的短期压力也更大，因此体现在多数个体和组织上的行为方式往往是急事优先。千里之外的中国团队主要通过邮件交流，邮件给人的感觉往往没有当面交流来的急迫和压力大，所以中国团队的问题通常不会被优先处理。

能否创造一个易于谈论要事的环境？能否让中国团队的请求产生同等的急迫感和压力？经过不断的尝试，我们发现利用Skype建立无间断视频（Always On Video）对改善交流问题有非常好的效果。中国和澳洲团队分别购置了大屏幕电视、摄像头、麦克风，注册了机器人账号，将Skype设置为自动接听、自动打开摄像头，专门用于在上班时的不间断联系。

不间断视频的引入大大减小了距离产生的不便，让团队在分布式环境下依然可以高效地交流，快速地反馈。

化整为零的成果展示

每次迭代后，团队都会为澳方的产品经理、客户

代表进行成果展示，演示在当前迭代中团队都完成了哪些功能，客户代表和产品经理会对特性进行验收及对产品本身和团队的运作状况进行反馈。成果展示会议通常会持续1~2个小时，但由于参与人数多，在时差的影响下会议时间非常难以协调，经常会出现关键人物无法到场或者会议无法按期召开的情况，从而使开发团队积压了很多没来得及验收的功能，既影响了开发进度又影响士气。

既然2个小时的成果展示会议难以预定，化整为零会怎么样？如果同时协调中澳双方的时间比较困难，那么引入在澳洲的代理人如何？

团队在回顾以往的会议后把这些想法提了出来，在后面的迭代中，团队在每个功能完成后立即向澳洲的负责人进行成果展示、收集意见，这样每次验收只需要10多分钟。在无间断视频的帮助下，团队可以随时发现负责人是否在场，利用各种碎片时间召开会议。

在迭代结束后，澳洲的负责人对各个特性已心中有数，他代表中国团队与所有相关人召开更大规模的成果展示会议，因为同样身在澳洲，组织会议就更方便、灵活。

通过化整为零以及引中间人，团队有效地减少了无法验收的功能，缩短了反馈周期。

持续发布与自动化

尽快发布、尽早发布是REA在线业务的客观要求，也是REA与ThoughtWorks双方所相信的文化。尽管双方都希望能够每日发布，甚至按功能发布，然而现实的种种困难让双方不得不妥协为每两周发布。

发布是一个需要多方共同参与、协作的过程。在我们的分布式环境中，产品环境的管理、维护以及产品部署是由澳洲的运营团队完成的。但部署过程通常并不顺利，一旦出现技术问题就需要中国团队来排查和解决，有时甚至需要三方甚至四方会诊。环境的不可控、远程定位问题的困难、高昂的交流成本，三小时的时差都增加了频繁发布的痛苦，让双方妥协为每两周“痛苦”一次。

从根源上分析，痛苦主要来自于以下几方面。

- 知识传递的困难。与测试、备份环境相比，产品环境复杂度更高也更敏感。学习这些知识是个长期过程，运营团队不放心把产品环境交给开发团队进行测试和部署。但由于不了解产品环境，又会引起开发团队出现各种纰漏，比如遗漏更新部署脚本。
- 手工过程让运营团队成为瓶颈。最终的产品部署是半自动化半手工的过程，每次发布都需要协调运营团队参与，手工过程既费时又费力让运营团队无法保证高频率的投入。
- 手工过程增加了交流成本：因为手工过程容易出错，一旦出现问题就需要双方在一起排查，时差和分布式环境增加了交流成本。

我们能否把在发布中需要交流的内容内建、固化在自动化脚本里，从而降低交流成本，提高效率和正确性？我们能否让发布过程无痛苦，从而能更频繁地发布？

经过摸索，团队使用了Capistrano和Chef通过Amazon的节点搭建出整套的部署环境，同时利用Go构建了持续发布的流水线（如图1所示）。

在这个过程中，产品环境的不一致性被一一修复，运营团队把产品环境的维护策略固化在Chef脚本中，让开发团队能很方便地使用。开发团队利用运营团队给出的脚本在Amazon上搭建出模拟产品环境的测试环境，在尽量真实的环境中进

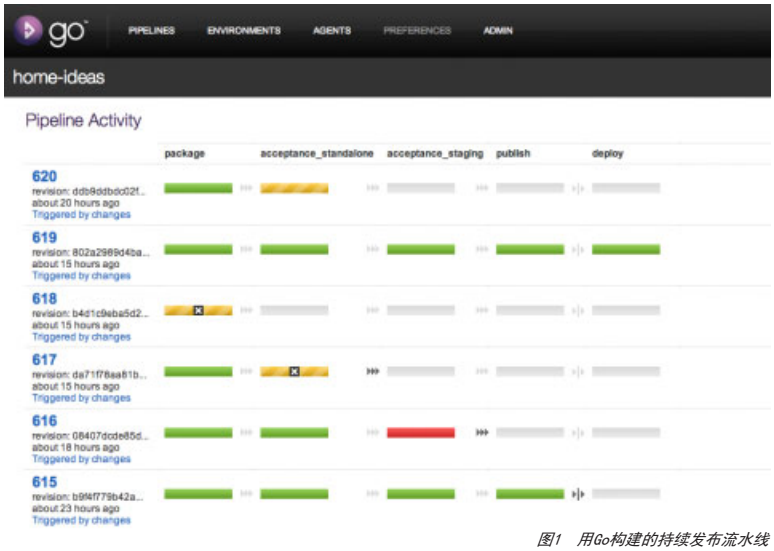


图1 用Go构建的持续发布流水线

行测试，以减少未来上线的风险。

目前的发布过程是：

- 单元测试、静态检查、覆盖率、打包；
- 运行单节点上的功能测试；
- 通过Chef在Amazon节点上创造出最小集群环境，在其上运行集成测试；
- 将通过测试的产品包发布到库中，供后续的探索性测试及更大规模的集成测试使用；
- 自动部署在产品备份环境中等待A/B切换。

同时我们做到了在任何时刻测试人员都可以用一条命令取下最新的发布包、构建出集成环境，展开探索测试。

现在，从提交到发布就绪耗时约1小时，而整个发布时间则不超过15分钟，做到了每日发布。

低耦合与高耦合

在合作初期，ThoughtWorks和REA都倾向于由中国团队开发比较独立的系统，减少分布式环境的负面影响。独立系统与现有系统的低耦合性，意味着相关责任人少、交流成本低、集成难度低、目标清晰、责任明确，同时独立系统即便出错，影响的波及面也比较小、风险易于控制。

随着合作的加深，独立系统的弊端开始显现。独立性使得双方的合作难以在更大的范围引起关注，缺乏了工作平台，就很难使产品架构师、测试架构师、运营团队和开发团队形成强烈的责任感和团队意识，因此组织间多点对多点的关系很难建立。而集成点少、集成难度低意味着对现有遗留系统的理解不足，难于承担更复杂、更重要的工作，团队很难扩张。

在后续的项目中，双方注意到了上述问题，意识到与现有系统耦合度高的新系统对加深双方的合作是有积极作用的，从而有意识地进行了投入。双方团队一起设定业务领域 的上下文、制定发布计划、确定技术路线，利用上文中提到的种种实践紧密地工作在同一个代码库之上。

这种工作方式不可避免地增加了出现问题后影响的波及面，在我们团队的构建脚本中有个任务叫prePush，它会运行所有的单元测试和部分关键的功能测试。而在一段时间后，中国团队认为对代码已相当熟悉了，就放松了要求，结果是构建失败后一下影响了澳洲3个团队无法提交代码。

这件事情发生后我们才发现代码库远比想象得复杂，我们对遗留系统的理解确实不到位，分布式环境会大大增加修复构建的难度和压力。这迫使团队在执行集成实践时更具纪律性，严格执行失败构建不过夜。为了能尽早地发现和处理问题，团队还开发了基于Jenkins的插件来让构建结果更醒目。

有些特别重视结果的团队还用上了指示灯，如图2所示。

图2　指示灯

对错误的及时响应，一方面缩短了故障时间，规避了开发高耦合系统的弊端，另一方面很好地建立了双方互信的关系。

隐喻

分布式环境也会增加建立隐喻（Metaphor）的难度。所谓隐喻是指在项目中的每一个人都使用相同的表达方式来解释同一个概念，表达方式可以是语言或图表，概念可以是项目目标、架构或业务概念。譬如“成为中国的Google”就可以是一个项目目标的隐喻，它往往通过牺牲概念的精确性来增加概念的流通性。

学习隐喻通常是在使用中进行的，通过反馈、矫

正和调整与他人保持一致。譬如在星巴克点咖啡，中杯代表着最小的杯子。与咖啡师交谈或者点错杯的经验能促使我们在下次点咖啡时使用正确的隐喻。然而在分布式环境下，由于不同的文化和语言背景往往会形成多套不同的隐喻，让交流变得很困难。在团队刚建立时，我们的团队成员有的用滑动图（Sliding Image），有的用大图指代首页上的一张图片，结果澳洲团队常常不明白中国团队在谈论什么问题，后来我们发现它的术语叫做主图（Hero Image）。

类似的问题不只存在于用户界面的组件和元素中，在技术词汇更为通用的项目架构、部署模型、持续构建流水线中，也会出现隐喻不一致的情况。这些不一致的隐喻不仅会增加团队沟通的开销，也会体现在代码和设计中，以错误的命名、不合理的抽象等形式降低设计的表现力。

通过摸索，我们发现术语表和可视化是一个比较经济有效的解决思路。

有了术语表，我们再和澳洲的用户体验设计师沟通界面设计、和架构师讨论技术架构，或是和运维人员商量部署方式时，因为系统隐喻不一致互相不理解的情况大大减少。利用术语表上的术语来进行命名，也让数据库表结构和代码的表现力更强、更容易理解。

除此之外，REA经常邀请中国团队到澳洲工作4~6周，在浸入式环境中快速掌握系统隐喻，中国团队也不断邀请REA团队来到中国，更高效地达成共识。最近，我们开始尝试“101视频”：由澳洲的资深工程师为每个复杂系统录制一段5分钟的视频，进一步降低交流的成本，统一词汇，提升学习效果。

为了让双方能更好地合作，让团队对业务领域有更多了解，双方还设计了很多其他活动，比如把REA的内部活动Townhall（向所有员工公布上一段时间公司的经营状况及未来发展方向）搬到西安，称之为mini-Townhall，帮助中国团队及时更新业务上下文；有的团队会每周留出半天进行骇客日，团队在澳洲的负责人会把有趣的创新成果纳入到迭代计划中，最终成为产品的一部分，做到产品发展思路集体共有，通过上述种种活动尽量减少信息孤岛、减少分布式环境对于团队产生的影响。

修复XP

与REA一年的合作主要经历了以下几个阶段。

- 试水：通过高度独立性的系统建立信心以及明确初步的合作关系和合作方式。
- 扩张：通过紧密的合作与一些开创性的实践从6人发展到45人。
- 沉淀：从边缘的技术与系统向核心的技术与系统转移。

在这个过程中，REA和ThoughtWorks一起面对和解决了很多困难，摸索出一些让敏捷实践在应用场景更具生命力和适应性的方法。譬如无间断视频、化整为零的成果展示、同一代码库等。未来REA与ThoughtWorks还将致力于延续信任关系，一起发现问题，探索解决方案。

我们所做的就是XP的另一核心实践：修复XP。没有哪种方法能保证一定成功，从简单的方式开始探索、及时真实地给出反馈、保持交流的开放、尊重差异、勇敢地面对困难，推动变化，这是我们学到的最重要一课。

感谢在写作过程中我的同事索勤、刘尧、崔力强所给予的帮助，同时也对帮我审稿的诸多同事一并谢过。

胡凯

ThoughtWorks公司程序员、咨询师及西安公司负责人。SpringSource官方培训师，开源软件的爱好者和贡献者，社区活动的参与者和推动者，个人博客是http://iamhukai.com。

责任编辑：杨爽（yangshuang@csdn.net）

产品经理让创业基因更炽烈

文 / 陆龙

自立门户、创立一家属于自己的公司只是创业的一种表现形式，真正的创业应该是一种精神，一种生存状态，是开创一种新的格局。本文将重点讲述缘何炽烈的创业基因可以让产品经理的工作更加卓越。

又到一年毕业时，不少意气风发的毕业生陆续地走入了职场，分配到了不同的工作岗位。我所在的公司也迎来了一批来自各地的应届生，其中就有一些被安排在了各部门的产品部，头衔是“助理产品经理”，顾名思义，就是帮产品经理做一些辅助性的工作，打打下手。

无一例外，这些风华正茂的年轻人，个个摩拳擦掌、踌躇满志，期待通过自身奋斗早日甩掉“助理”二字，这样就能堂而皇之地以“经理”自居了。可多数人却不知道怎样成为一名合格的产品经理。

确定方向

产品经理到底是什么呢？网上那些关于产品经理定义、职责和能力模型的文章，相信很多新手已烂熟于心了。这些文章确实能给刚入行的产品经理许多指引，同时“创业”这个词也让即将成为产品经理的人感到一丝兴奋和不安，尤其是在移动互联网如此盛行的今天，创业似乎成了产品经理必然的宿命，网络上流行的各种文章或者访谈语录，仿佛在铿锵有力地告诫着每一个产品经理：“不创业枉为产品经理”。

然而，常被我们津津乐道的创业，从来都不是一个信手拈来的轻松话题。创业家今天看似风光无限，但其中艰难曲折又有谁能知晓。各种创业励志的图书很多，但仔细想想，又有谁的成功可以被轻易复制呢？乔布斯曾经说过一句话，让我印象深刻：“You've got to find what you love”。相信很多刚刚入行的产品经理，在悉心研读各种关于产品经理能力和素质培养的文章，这并不是坏事，但更重要的是，得搞清楚自己爱什么，并找到自己的方向，这才是一切事业或工作出发的基础。而对产品经理而言，只有自己的方向明确，才能定义产品的方向，也才能指导团队方向，甚至公司的方向。从另一方面来说，一旦确认方向，在朝着它前行的过程中，很多东西都成了必然的副产品。比如，产品经理这个头衔、产品经理优秀的能力和素质，或者是一家公司。

创业是种生活状态

也许有人会说，我初进职场，哪能一下子就搞清楚这么多东西？确实，对于刚入行的产品经理来说，这个要求确实有些高，然而古人说得好：凡天下难事必化为易，凡天下大事必作于细。那么，作为一个助理产品经理而言，首先应该解决的问题是什么呢？在一个部门里，无论是助理产品经理还是产品经理，都应该是最了解产品的人，大到产品规划，小到界面细节，事无巨细。你一定要充分了解产品，让自己成为其真正的“超级用户”。在以产品为主导的公司里，产品经理应该是最理解公司产品战略的人之一，能够站在老板的角度

思考产品的问题，甚至在一些细分领域或者细节上，要有比老板有更深刻的理解，这是比较高的境界和要求，当你达到这样的境界时，你会忽然发现，其实你离“创业”并不遥远，因为老板就是通过产品去创业的人，这难道不是每个产品经理梦寐以求的事情吗?

因此，很多时候，我所理解的“创业”，并不一定是自立门户，独闯单干，成立一家属于自己的公司。这只是“创业”的一种表现形式。创业应该是一种精神，一种生活状态。当你是一名员工时，如果你拥有了这种精神，那么就能找到自己的职业方向，并且把工作当做事业来做。有句话说：“心中有佛，处处是庙”，如果你拥有这份精神，找到了心中的“佛”，“庙”在何处、是什么样子又有什么关系呢?

我认识一些非常优秀的同行，每隔一段时间，他们就会把移动应用市场里排名前100名的应用都研究一遍，无论花多少时间和精力。甚至很多人会左手Android手机，右手iPhone，时时刻刻都在琢磨和钻研各种应用。因为对产品工作的热情，甚至激情，才能造就合格甚至优秀的产品经理。

优秀的产品经理总是大受欢迎，因此，很多公司对员工、尤其是产品经理都提供内部创业的机会。以我所在的公司为例，老板为了留住优秀的产品经理，发挥产品经理的最大价值，在公司管理的制度和文化上，给予产品经理足够的空间，充分授权，使其能够在公司内部进行创业，定义产品，组建团队，从而进行相对独立的项目开发，这也是所谓的定方向、建班子、带队伍。反过来说，也只有具备了创业精神的产品经理，才能在这种制度和文化下得到最大程度的提升和发展。

开创新格局

当然，以产品经理的角度去思考创业，首先还是要解决产品的问题，拿不出产品主张的产品经理，没有谈创业的资格。产品经理最大的价值在于产品创造力，他需要去定义产品的方方面面，比如，这款产品是什么，从哪里来，最终要到哪里去，并且负责产品整个生命周期的成与败，这是一般意义上产品经理的职责所在。而创业，其实是开创一种新的格局。这种格局可能是某个具体产品的功能或界面设计的优化，也可能是某个新产品的创意策划，更有可能是具备搭建整个产业生态链能力的产品线方案，格局无论大小，贵在你是否像“疯子”一样的投入，并最终产生价值。只有依托产品去开创新的格局，才是产品经理创业的根本。

很多业界前辈通过一两款成功的产品，开创了自己的公司。但无论产品多成功，他们都会低调地声称自己只是公司最大的产品经理而已。这是对产品经理价值最大程度的认同，也正是这样，它激发了无数人对产品经理岗位的无限向往与憧憬。业界这种对产品经理价值的高度认同，缘于行业竞争的加剧，尤其是在互联网软件领域。产品能否通过完善的功能和出色的应用体验抓住用户，决定了一个公司的生死，而产品做得好不好，很大程度上要看产品经理。从某种意义上来说，互联网软件公司的管理其实都应该是围绕产品展开的，这种围绕产品的管理将给产品经理更大的空间和资源去发挥和创造，这是所有互联网产品经理的机会和挑战，作为产品经理的你我，这真是一件令人激动不已的事。

在产品经理的眼里，这是一个由各种各样产品组成的世界。产品经理设计和创造着各种各样的产品，去满足不同人的需求，完善着这个产品的世界，因此，产品经理也就拥有了这份权利和义务：用产品改变世界。无论你是新手菜鸟，还是老江湖，创造一款能够改变世界的产品，应该是每一个产品经理的终极梦想，“Make a dent in the universe”，这也是我对“创业”的深层次理解，即开创新的格局。而这不只需要各种能力和素质的支撑，更需要的是你的骨子里有炽烈的创业基因，它也许燃烧的没有干柴遇到烈火那一瞬间猛烈，却能持续地发光和发热，照亮每一个产品经理的创业之路!

陆龙

就职于深圳万兴软件，历任万兴数款桌面软件产品经理，现专注于移动互联网。苹果迷，手机控，偶尔玩玩网游，喜欢不断尝试新的东西，当然只限于互联网产品领域。

责任编辑：杨爽（yangshuang@csdn.net）

技术引领体验

网易邮箱技术总监向东专访

记者 / 长卿

网易邮箱附件上传功能近日完成升级，简化上传流程，用户可直接发送最大2GB附件，无须选择上传的附件是普通附件还是超大附件，无需安装任何插件，整体实现提速翻倍。记者就极速性能、智能识别等功能的技术实现等问题对网易邮箱技术总监向东进行了专访。

修改既有附件传输协议 采取全新分布式上传

《程序员》：网易邮箱如何实现整体提速翻倍的？

向东：我们通过在北京、广州和杭州等地大量增设服务器，根据国内的网络环境优化了在各大运营商的服务器布局。用户通过浏览器登录网易邮箱上传附件发送邮件时，系统将会自动定位并连接到离该用户最近的服务器。网易邮箱在实现统一普通附件和超大附件上传入口的同时，也实现了附件上传的提速，邮件所带附件越大，发信提速越明显，整体提速翻倍。

《程序员》：网易邮箱为上线新功能，修改了既有的附件传输协议，采取全新分布式上传系统，能详细谈谈吗？

向东：以往邮件系统通讯使用的是标准的SMTP以及POP3协议，由于协议的限制以及其它邮件系统的限制， 往往发送附件的大小最大为50MB，目前网易邮箱在既有的传输协议，进行了扩展，使之兼容目前现有的邮件协议，又可以支持发送2GB的附件，并确保包含2GB大的附件的邮件能与其他的邮件系统互通。

兼容无插件直接上传、HTML 分块上传等多种方式

《程序员》：网易邮箱是唯一具有2GB超大附件和普通附件统一入口的产品，这个“智能识别”功能的实现有哪些技术要点？

向东：这个功能的实现主要包含四个关键技术。

第一，系统自动识别当前邮箱类型，从而判断出最大可支持的普通附件大小，智能选择应作为普通附件上传还是云附件上传，并且根据用户添加的附件大小自动更改附件类型，无须用户人工干预。

第二，前端统一代理自动兼容普通附件和云附件两套上传协议，使用户能得到一致的上传体验。

第三， 兼容无插件直接上传、HTML5方式分块上传、插件上传、Flash上传等多种方式，智能判断客户端的环境自动选取最适合的工具进行上传。

第四，在安装了插件的情况下，支持闪电上传，判断若云端已有相同文件，则无须重复上传。

最大发送2GB“云附件”

《程序员》：请解释下“云附件”这个概念？

向东：“云附件”的概念主要是指用户今后不需要再理会附件存储的具体位置， 可以通过各种终端无差别地对“云附件”进行访问和各种操作。网易邮箱云附件的优势之一就是用户无需安装插件就能上传2GB附件，这个给网络环境相对较好的用户带来了极大的便利，另外网易也针对网络环境相对较差的情况进行了优化，通过安装扩展插件，用户体验能得到更大的提升。

《程序员》：现在，电子邮件它不再仅仅是种消息传送协议，集成越来越多的功能，如E-mail、日程、通讯录等，网易邮箱是否有针对变化作出改变？

向东：目前我们主要从五个方面进行了准备。

第一，支持更丰富的通讯录选项以及内容，支持vCard 3.0标准；

第二，提供丰富日程管理功能；

第三，更快速的通讯录、邮件、日程同步，更全面地支持现有的各种同步协议；

第四， 把网易邮箱的功能更好地集成到各种终端设备上，方便网易用户使用；

第五，进一步改善邮件上传和下载的速度以及可靠性。

例如，网易邮箱日前已全面支持Exchange协议，可以实现邮件、通讯录和日程管理三大功能的同步，用户在电脑上收发邮件、管理联系人、安排日程都会同步到手机和平板电脑上。此前网易邮箱已在苹果的iOS和Mac OX设备上提供了日程同步和通讯录同步功能，并提供了一键配置的快捷方式，大大地方便了苹果用户使用网易邮箱的相关服务。

系统智能判断实现闪电上传

《程序员》：对于邮箱用户而言，“用户体验”最重要体现在哪几个方面？

向东：本次合并附件上传入口，用户感受最明显的地方主要有两点：一是兼容普通附件和超大附件，用户无须干预；二是可以支持免插件上传超大附件，简化上传流程。

网易邮件事业部总经理柳晓刚表示，网易邮箱始终致力于提升速度和改善用户体验。此次附件上传全新升级，不但能够大大改善用户在附件上传体验，而且附件上传速度也显著提高。作为技术领先的邮箱运营商，2012年年初，网易邮箱全球率先支持语音搜索邮件；同年6月，网易邮箱又率先应用自主研发的人脸识别技术来提高邮箱的安全等级，目前国际互联网尚无在用户安全方面运用人脸识别系统的先例，这也填补了互联网领域的一个空白。网易邮箱也将技术创新作为使命，力求在未来给用户带来更多惊喜。

网易邮箱附件上传统一入口后，可以同时通过群发100个好友。对用户来说，最显著的体验就是写邮件时无需选择上传的附件是50MB以内的普通附件还是超大附件，也不必安装任何插件，直接上传即可，且附件数量不受限制。当用户上传的附件总容量超过50MB时，为避免接收方邮箱无法支持情况，网易邮箱系统会自动转为使用“云附件”的方式发送，以保证对方能正常接收。网易邮箱发出的超大附件，任何其他邮箱都能接收，不受对方邮箱空间大小限制。

目前国际主流邮件运营商Gmail、Hotmail等最大只能发送25MB附件，而国内邮箱运营商虽然也支持2GB附件，但需要用户自己判断附件上传方式， 有时选择错误需要再重新操作。网易邮箱实现统一的附件上传入口直接发2GB附件，为邮箱行业内首家，附件上传功能的体验大大提升。如果用户上传的“云附件”已存在网易邮件服务器中， 系统经过判断，用户再次上传该文件时则可以实现闪电上传，数秒就完成上传。

特别想说明的是，网易邮箱最近一年在附件上传功能上不断改进完善，陆续上线了附件上传提速、上线2GB超大附件、桌面文件拖拽上传功能和附件拖拽上传以及复制/粘贴式添加上传。网易邮箱用户在手机和iPad等终端（包括PC客户端、Android手机端、iPhone端、iPad端的）上在线预览附件等功能。

一分钟先生

如何做好技术布道

技术布道者是IT行业中的新角色，因此布道者的职责是什么以及如何能做好技术布道成为大家关注的问题。本期几位嘉宾将为大家揭开技术布道的神秘面纱。

冯大辉
丁香园技术团队负责人

用影响影响影响

在我看来，互联网行业先后有几个布道者值得效仿：前微软员工Robert Scoble，前Yahoo!员工Jeremy Zawodny（现在在Craigslist），Google的Matt Cutts，以及创新工场的蔡学镛。这些布道者，孜孜不倦地对外传递大公司的动态、新的技术趋势、新的产品信息以及他们自己对技术的思考感悟，有了他们，微软、Yahoo!、Google不再那么神秘陌生。当然还有一些无法效仿的技术布道师，比如互联网之父Vint Cerf现在是Google的首席互联网传道士，只可高山仰止。

“布道”这个词，颇有一些宗教意味，的确，如果不是特别热爱或是信仰的话，是做不了技术布道的。有一句话叫“用影响影响影响”，说起来有点绕口，这句话也可以用来形容“技术布道”这件事儿。技术布道者要能够建立起足够的影响力，用这个影响力去影响一部分具备影响力的人，再促使这部分人自发地去影响更大的目标群体。好的效果是潜移默化的。

一个好的布道者不一定是一个好的演讲者，技术会议上能够滔滔不绝地演讲当然更好，但如果拙于口才也无碍，只要你是一个好的写作者就足够了。我在前面提到的几位值得效仿的技术布道师，都具备足够相当好的写作能力，我甚至分析研究过他们的写作方法和技巧，他们的博客经常有内容更新。我一向认为，文字传播的持久性远比视频、语音的效果好很多。具备优秀的写作能力是成为一个优秀技术布道者的必备条件。

一个好的布道者要懂得利用新的传播媒介，他们一定是Twitter、微博的活跃用户，不懂得利用新传播媒介的人做不好这件事情。善于利用新媒介的另一个好处是，获得相同的语境后更容易赢取社区用户的信任感，也能更好地和一部分具备影响力的受众互动。

技术布道不是一项“工作”，建议最好不要弄什么KPI之类的进行考核，因为效果实在无法量化。某些商业公司为了使自己的产品能被更多开发者接受，派一大堆人到各种会议上演讲，那不是布道，最多只能是一种营销方式。

技术布道需要坚持和耐心，如果只把它当成一项短期的任务是无法做好这件事情的。还不如去开一次媒体发布会。

作为技术布道者，你自己也会得到一些其他的“收益”，比如，更加有名气，当然是虚名，参加各种会议上有人叫你“老师”或者“专家”了；也有负面的影响，不排除有人说你“忽悠”，所以心理素质要好才成。相信一件事情，你做的事情是有价值的。

张辉
百度移动云事业部高级产品经理

也来说说技术布道师

说起布道师和布道，有人觉得陌生，其实它距离我们很近；有人觉得忽悠，其实它很真实。

技术布道是随着IT产业的兴起而兴起的职业。与传统的“宣传推广”不同，它是针对特殊产品、面向特殊人群、采用特殊方法进行“宣传推广”。从性质上，靠近“Marketing”；从行为细节上，更靠近“技术”。

职责

一个立志做平台产品的公司，其布道师团队需要为以下几点负责。

- Awareness：平台的核心声音是什么？主要面向哪个细分的市场？以怎样的策略去表达这些声音？主要对一线开发者说还是对决策者说？这是布道的起始，主要是策略确定。
- Adoption：这是最重要的实施环节，需要通过各种技术手段，向目标人群布道，说服其接受、采纳自己公司的方案。这个过程不是一次性的，更多的是渗透进行。这个过程，充满反复的实践，也是大家最多看见的部分。
- Leadership：做技术布道的最终目的，是要目标市场相信自己公司在这个细分领域的技术领导力和权威性。

以上几点是布道师团队整体的职责，每名布道师，所承担的任务不同，有人更偏重于策略，需要的是技术理解力和Marketing Sense与背景；有人更偏重于实践，需要有技术背景，尤其是服务于一线开发人员的技术布道师，必须是技术出身，能以代码为基础向开发者宣讲技术（Apple的JD中主要就是描述这种职位）；而能确立一个公司技术领导力的，则取决于少数几个人，比如Amazon CTO Werner Vogels，可以说，他的个人影响力是人们相信Amazon在云技术领域技术领导地位的重要因素之一。

入门门槛

对广大程序员而言，哪些人适合做这件事情呢？基本要求是“理解技术、根植于技术”，现一线做研发或者做架构师的人都有机会成为一名布道师。再往上走就是要表达能力突出，包括书面或者口头。因为这是一个需要通过反复沟通达到布道目的的过程。自己理解和让其他人理解是两个层面的事情。如果还停留在“茶壶里煮饺子，倒不出来”的阶段，那是得好好锻炼一下了。

晋级标准

技术素养和表达能力是基础，而对布道师而言最重要的就是信仰的力量+传播的热情与技巧。很多人一听到“布道”，不免会立马想到“忽悠”一词，其实布道最重要的不是让其他人相信，而是自己相信。这种相信不是无条件的，也不是一开始就有的。很多时候，是自己先怀疑、否定，然后经过实践、学习、探讨、思考等否定之否定的阶段，最终达到深信不疑。

很简单，如果我们自己都不信，凭什么让别人相信？乔布斯为什么那么具有说服力，因为他有《乔布斯传》中所称的“现实扭曲力场”，因为他“信”，甚至他的死都与自己的信仰有关。这就是布道师，这才是布道师。

接下来要做的就是去传播，这需要热情与技巧。关于热情与技巧，大家最好的了解方式就是参考大师的布道作品。

未来，随着平台化越发成为趋势，平台化越发成为一种流行的商业模式，对于布道师的需求也就越发强烈。因为这个职业需要天然的热情+综合的技能+知识的沉淀+人脉的积累，预计这会成为一个很好的发展方向。

白明
东软集团移动互联网事业部第一开发部技术总监

改善技术布道效果的几个实践

技术布道不易，想取得良好的效果就更难了。下面分享一下改善技术布道效果的实践。

■ 自我认知。技术布道前，布道者首先要做好自我认知，这将有助于他确定自己是否胜任此次布道以及采用何种布道策略以赢得更好的效果。认知的内容包括：自己是否精通这方面的技术，若只知皮毛，布道效果将大打折扣；自己在组织内部是何种资历与角色，如果你是职场新人，人微言轻，布道效果势必会受到影响。

■ 环境认知。布道者应根据对组织环境认知的结果来选择适合的布道策略。认知的内容包括：组织内成员是否拥有一个开放的心态，乐于接受新鲜事物；组织内部是否为大家建立起一个有影响力的布道平台并设立奖励机制；管理者是否积极支持技术创新并接受因此带来的成本损耗。

■ 精选主题。技术布道的主题无疑是影响布道效果的最直接因素，因此在选题时要把握住一个至关重要的原则——主题一定要能够给组织带来价值。这些价值可体现在多个方面，例如解决困扰大家已久的问题、提高工作效率、降低风险或增进理解等。从我多年布道的经验来看，从问题出发选题是个不错的选择。而且因为这类主题与大家的日常工作息息相关，你可以相对容易地获得良好的布道效果。

■ 受众分析。技术布道的主题多数并不具备普适性，它只是在一定受众范围内是有影响力的，因此在谋划布道之前要做好布道受众的分析，识别出适宜本次布道的受众。一旦确定了受众范围，布道者就可以在布道之前对受众遇到的问题进行相关的分析和调查，使得布道更有针对性，可取得事半功倍的效果。

■ 把握时机。俗话说：“来得早不如来的巧”。技术布道也是一样：“布得早不如布得巧”。良好布道时机的把握对赢得良好布道效果有着很大影响。如果你非要向一个下周就要做产品发布的产品线推广JUnit，显然你选错了布道时机。人家都忙得脚打后脑壳了，你还给人家添乱。

■ 制定策略。制定一个适宜的布道策略对取得良好布道效果作用很大。

■ 以点及面。受众面越大，布道的效果可能越不理想。因此，最好先在小范围内进行试点，并在取得成果后再扩大布道范围。“事实胜于雄辩”，小范围布道取得的成功结果会让更多人见识到该主题的价值，并带着更加积极的心态参与到这个主题的后续布道中去。这点在说服上层管理者时也尤为有用。

■ 划分阶段。如果布道主题涵盖范围较大，可将布道划分为多个阶段并逐段实施。每实施一段后，可以根据受众的反馈进行自我调整和完善，这有助于在下一个阶段布道中取得更佳效果。

■ 善于借势。如果布道的主题在实施之前获得了管理层的认可，那不妨将布道名正言顺地打上官方烙印。相比于职级对等的“水平布道”而言，借管理层之势的“垂直布道”势必更能引起大家的关注，提升大家参与的积极性。

■ 心理建设。大多布道者对技术充满热情，对布道乐此不疲，希望通过自己持续不断的布道使得组织获得能力提升。但布道的结果有成功也有失败，失败时的苦果并不容易下咽。因此布道者在布道前先要做好心理建设，最大可能地减少布道失败对自己的负面伤害。

■ 保持耐心。只要你认定某种布道会给组织带来价值，那么就不要放弃，要有耐心，充分考虑使用上面所述的策略和方法，坚持做。

■ 降低目标预期。这是个心理把戏，在布道前适当降低些目标预期，即使布道效果未达到你的期望，带给你的伤害也不至于很大，有助于你保持持续的布道热情。

马林
人人网高级布道师

我能成为一名布道师吗

一名优秀的技术布道师，应该是这样的：在技术人员眼中，他是市场高手；在市场人员眼中，他是技术牛人；在会议听众面前，他是充满激情的演讲者；在合作伙伴面前，他是经验丰富的咨询师；在产品经理面前，他是把握市场需求的建议者；在创投对接的场合中，他是熟悉商业模式的引路人；在同事面前，他是永不疲倦学习新技术、了解新产品的先行者。

成为这样一位布道师，需要具备哪些特质呢?

你必须是个技术牛人！

我所熟识的布道师，无一例外都是技术高手，而且大部分人精通多门编程语言（有些已退休的布道师甚至是COBOL程序员出身，随后转战C/C++、Java、PHP/Perl/Python或是Ruby）。由此可见，技术能力尤其是学习新技术的能力是技术布道的基础。

只有外向者适合做布道师？错！

印象中的布道师多是在讲台上侃侃而谈的家伙，似乎性格外向的人更适合这个职业。其实，平时略显沉闷但往往语出惊人的布道师大有人在。这好像不符合常识，但借助MBTI人格分类可以解释这种现象。

技术演讲带有强专业性和目的性。专业性不必详述，目的性体现在：布道师不仅要说服听众采用你所推广的产品或技术，而且要避免简单和生硬。同时，在欧美一些技术市场工作已成体系的企业中，早已有诸如销售线索数量、销售收入、销售转化率等量化指标去跟踪和衡量布道工作的结果。

因此，为说服听众、达成目标，详实的论据和内容安排的技巧就成了布道师准备演讲时思考的重点。根据MBTI理论，内向者擅长从独处中获取能量，反而更适合进行这样的思考和准备；而外向者需要寻求和他人进行交流，在谈话中获取能量（一部分极端的外向者在被迫独自思考时，甚至会不自觉地自言自语起来）。

可见，性格并不能成为技术布道的阻碍。外向者和内向者通过有意识的训练，都可以弥补自身缺点，成为优秀的演讲者。

你是一名“极客”吗?

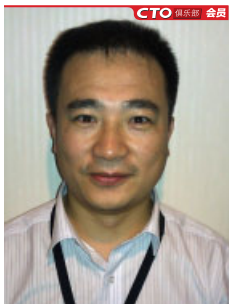
这里的极客，是指热爱探索和创造；痛恨因循守旧但不自怨自艾，而是信仰技术的力量并动手去加以改变的人。最近流行的“降级论”正是一种用极客精神去改造传统行业的体现。

极客和布道师有着天然的联系。

供职于平台型企业的布道师的职责之一就是针对公司的新产品进行技术市场推广。因此，你通常是新产品的最初用户，也是最先精通新产品的一批人。你要在资料匮乏的情况下，摸索产品特性并制作课件、视频，撰写文章；你要代表第三方开发者去拷问产品团队，产品是不是好用，需求有没有满足，是不是解决业务问题；你要代表公司去面对开发者的疑问甚至竞争对手的FUD；除技术之外，你要怀着一种热爱去探索和开发自己在市场营销、演讲技巧、商业甚至财务等方方面面的技能。

如果说布道师带有信仰意味的话，我更倾向认为这是一种对“我所做的事情一定可以带来某些变革”的信仰，而不是无条件地相信新产品本身。抱着这种与极客精神一脉相承的信仰，你才能有持续的动力去发掘产品中的兴奋点，也才能推动产品不断修正和演进。

技术布道并不神秘，对入行已久、经验丰富、希望自身有所转型和突破，但又不想放弃多年积累的技术人员来说，布道师不失为一个理想的转型目标。



对外布道，布道者代表着整个公司的技术实力，他们的工作重点在于宣扬公司产品和技术理念，获得客户认可。而组织内部的布道者的目的是推动组织的技术发展，使组织的技术结构日趋成熟和完善。当组织对布道采取积极鼓励的态度，对于增强布道者的信心和积极性将起到非常大的作用，布道的过程将会较为顺利。相反，如果组织本身缺乏技术创新的氛围，那么布道的过程将会是困难和艰辛的。布道的效果最好能在短期内呈现，否则受众的信心和兴趣会随着时间的推移逐渐丧失，推广的过程也就变得越来越艰难。如果只是为了引进新技术、为了让团队看起来比较酷而进行布道的话，那么这样的布道是没有生命力的。

——孙力策，海和信息技术（大连）有限公司总经理

技术布道需要的是不加保留。入行不久的技术人员又是会遇到别人没有耐心的指导。一方面可能是问的问题太简单，别人认为利用网络会很容易解决，另一方面可能是受“教会徒弟，饿死师傅”这一说法的影响。其实这种情况在传统行业可能会有，因为传统行业的技术内容有限或创新有限，在IT行业，永远都是学无止境，在教会别人的同时，自己也能有更深入的认识。要注意从受众的立场出发，如果是设计人员，可能需要掌握技术的利弊和实现思路；如果是编码人员，可能想要了解具体实现，不同的岗位侧重点不一样。

——周庆松，北京美特软件技术有限公司技术部经理

技术布道可以检验自己对技术的掌握程度，在使用技术时，可能不需要对它的边边角角都很清楚，但在向别人解释时就不一样了，听众的问题经常会让人意料不到。如果你的回答是“解决了这个问题，剩下的事情就简单了”，那就需要反思一下自己是否真的理解了“这个问题”的轻重程度。如果“剩下的问题”真的很简单，那为什么不索性解决“这个问题”，一鼓作气给一个完美的解释呢？有时，不仅“这个问题”不简单，而且解决了“这个问题”后面还有“那个问题”——“正入万山圈子里，一山放出一山拦”。我认为，所有讲不清楚的地方都是源于理解不清楚。

——施懿民，上海知平信息技术有限公司总经理

技术布道者一定要对自己所推广的技术、工具或新方法的价值点了然于胸，通过选取恰当的小范围试点和一定的试错经历，最终发现价值点和顺利实施的方法。在试错过程中解决的问题和产生的价值可以很容易与其他团队成员产生共鸣，那么技术布道的成功实施就很容易了。而且要注意，数据是最具说服力的“武器”之一……通过详细的调研，对问题有深入的了解之后，“用数据说话”，通过制定详细的实施计划以及计划能产生的效果，将更容易获得认可和支持……一件事情放在一个团队或者组织的不同阶段都会显现出不同的轻重缓急，选择合适的时机也是非常重要的。

——靳松波，南京班墨自动化技术有限公司创始人

移动产品用户体验与设计系列（二）

接地气

文 / 林敏

不久前在北京参加一个活动，其中有某研究院的一个分享。我本很期待听到、看到该研究院在帮助公司提升竞争力方面所进行的努力，但等到的却是资历丰富的研究员颇带自豪地分享着华而不实的项目，着实令我错愕。当真是一个象牙塔中的理想国，大象已站在悬崖边上却视而不见，关心的是如何让大象跳舞而丝毫不考虑如何让大象离开危险。

无独有偶，出差期间还听到发生在另一家企业的故事：一位一心追求极致体验的年轻人跟公司要求以自己的方式做一款产品，获得了支持。于是年轻的产品经理开始搭建自己的团队，招募了不少设计师，人数比工程师还多，同时要求工程师在设计主导下进行开发。这个团队成为鲜有的完全设计驱动的团队。产品上线后，用户群体小众、用户量增长缓慢。但年轻的产品经理不以为然，抱着持有用户便是价值的思想，惨淡经营，终于落得项目停摆、团队解散的结果。

前面两个故事中，一位是资历丰富的研究员，一位是身怀抱负的年轻人，都出自知名上市大企，却都犯下相同的错误：不落地，接不上地气。这本不是什么新冒出来的问题，但值得思考的是两个故事里的核心人物都是忠实的用户体验追随者，而这才是最最危险的信号。

用户体验这两年日见红火，但口头价值远超实际价值。从某种程度上说，用户体验在国内也处在落地不稳的状态。虽然各大公司都有自己的用户体验团队，甚至单独成立部门，但做得好的仍在少数。许多本土企业在用户体验方面的投入一直顾虑良多，也都是对不接地气心存忌惮。曾有机会同时和某知名本土企业负责设计和负责研发的两位高管一起聊用户体验，他们都不约而同地表达了对用户体验价值如何体现的关注，都希望先看到兔子再撒鹰。

因此，用户体验接上地气就显得尤为重要。这就要求设计团队不能够只懂得设计、只关心设计、只满足于设计。在设计之外，还要懂研究、懂技术、懂生意。这些都是让用户体验能够落地的重要保障。

研究是对用户的保障，除了证实需求的真实可靠，同时也明确了目标用户，可以有效避免设计师自己扮演用户的情况。技术是对实现的保障，让需求不再是空中楼阁，也为设计的取舍与平衡提供有效的输入。生意则是对可持续的保障，套用刘强东的一句话“每个不赚钱的公司，都该感到羞耻”。

不少设计师都问过我类似的问题：为什么我们考虑产品创意时要关心技术，关心怎么挣钱？这个问题一听挺有道理，不是还有研发团队和市场团队嘛，有他们不就行了。这在过去传统的软件产品开发上是可行的，因为周期长，可以一个一个团队完全分割清楚来做。但在快速变化的今天，再这么做就无法跟上市场的速度了。设计师不在输出创意时就已对技术对生意有思考，那么这些创意能够接上地气的概率就会非常低，其结果是浪费企业资源。出现得多了，设计团队就会由于挫折感的增加而失去活力，企业的设计竞争力迅速下滑。这却很不幸地成为不少企业的现状。

接地气，是当下用户体验团队的第一要务。

林敏

用户体验践行者和布道者，曾创立UU网为国内早期从业人员提供学习交流平台。现为三星中国设计研究所用户体验创新部负责人。

责任编辑：陈博（chenbo@csdn.net）

《酒店达人》设计谈

文 / 刘张博

《酒店达人》的成功绝非偶然，其创始人刘张博在此文中以时间线为指引，将《酒店达人》的设计体验升级之路娓娓道来。

《酒店达人》的产品灵感，来自一次我自己找酒店的痛苦经历。

2010年，北京，一个炎热夏日的午后，我拖着行李箱走在清华大学东门外的一条小路上，要临时更换一家附近性价比合适的酒店，为应对此问题我已提前在我的Android手机上装了携程的客户端应用。但使用起来却迟迟没能帮我找到合适的酒店信息——由于其功能当时仅为互联网数据业务在手机端的平移，酒店展示方式仍是传统PC互联网的列表形式，只能按照价格、星级或评价排序，但在移动的场景下，无法搜索到跟我具体位置相关的酒店。因此，我决定自己做一款能够随时随地查询周边酒店的应用。

位置 价格的信息展示方式

接下来就进入到产品的最初设计和构思阶段。我们在确认用户需求重点时仍延用了我当时的真实需求。

- **位置**。我需要找的是我周围的酒店，通过步行即可到达。
- **价格**。我是价格敏感型用户，当时我期望的酒店价格在200~300元之间，超出这个范围的酒店不予考虑。

我认为以上两点就是用户主要的需求点，所以我们设计了一个以地图为主要界面的交互模式，在其上添加酒店的位置和价格标签。如此一来将位置和价格这两个用户查询酒店时最关注的条件以非常直观的方式结合在了一起，供用户参考。这个设计使得查询效率得到了非常显著的提升，在手机上大大地减少了用户从查询到决策的时间，同时也把从查找酒店到预订酒店的时间缩短到以分钟计。

我认为对该功能的首创是《酒店达人》iPhone第一版在“零市场推广”的条件下大受欢迎并很快登上苹果App Store旅行类榜首的主要原因。它真正切中了用户需求痛点。

确定了以用户所在地为中心进行搜索和展示的主要目标之后，需要回答另一个问题：到底显示用户附近多少家酒店？10家、50家还是100家？经过反复比较和尝试，我们最终根据用户所需的信息量及手机屏幕上一次可以清晰显示的酒店标签数量，决定最初显示离用户所在位置最近的20家酒店，这既为用户提供了足够多的选择，又有效避免了信息过载。同时，也让有需求的用户可以选择查看“更多酒店”。

长按设定任意查询参考点

解决了以用户所在位置为参考点的搜索之后，设计思维的惯性很自然地把我们带入了下一个进阶问题：如果用户想提前预订自己尚未到达的位置附近的酒店怎么办？这个问题在当时屈指可数的

几个同类型客户端里都没有得到解决，但我们相信用户一定有此需求，并且此需求的满足一定能帮助我们提供更好的产品使用体验。

确定了目标后，我们开始构思实现方式。主要的难点在于操作方式的确定，需要考虑的问题有以下三点。

- 地图本身已有很多的默认操作方式，我们新加入的操作不能跟其中任何一个常用的操作冲突。
- 这个操作方式必须足够简单易上手。
- 用户能主动触发，而非平移地图时自动加载。

经过不断尝试，我们发现“长按”是最适合我们的操作：首先它没有同地图操作中最常用的平移和缩放操作相重叠，易于操作；其次它肯定是由用户主动触发。我们通过反复实验确定了一个长按触发时间的数值，并延用至今。

在确定了这个操作方式之后，只剩最后一个问题：如何知会用户？

弹窗最直接，却是一个很不友好的方式。大家经常会将应用中其他的一些小的使用技巧放在Loading状态框中，但长按操作的重要性又同它们不在一个层面上。因为该操作只能在地图模式进行，所以最好把提示放在地图模式内部，而且还要在足够醒目的同时又尽可能少地影响用户。

于是，我们在地图上部制作了一个浮动框，这个浮动框会随着地图的首次展开而下滑，这一个动作足够引起用户的注意。为了不长期占据被浮动框遮挡的那一块地图区域，在一定延时之后浮动框会自动向上滑动收起。这个解决方案堪称完美，从各个方面都符合我们的要求。事实上，随后我们的同行都认同并逐渐采用了这个方案，这个微创新也算是我们对这个细分领域所做的一点小贡献。

对某一房型只提供唯一的服务商

随着酒店达人在业界知名度的提升，逐渐有更多酒店行业的知名企业希望同我们建立合作。由此我们接入了多家数据和业务供应商，那么该如何展示它们的数据？

图1　对某一房型只提供唯一的服务商

哪儿网的模式是展示所有可选的服务提供商，包括报价和可接受的预订方式，但这个方式在移动端有两个问题：一是需要多一步操作，二是有的服务提供商其实不支持通过手机端在线下单，展示没有实际意义。

基于对这个问题的思考，我们决定对任一酒店房型只提供单一的服务商，为此我们在后台实现了一套算法，用于对我们的服务商进行排序。我们最主要的标准是帮助用户高效地完成消费决策。因此，我们做出筛选，为用户提供有价格优势，并且服务稳定、可靠的合作伙伴作为参考，避免用户自己再去面对众多庞杂的信息，客观上也使得消费者直接受益。

语音搜索

有一次，我们的一位合作伙伴跟我说，他出差在外时，单手使用他的大屏手机来操作包括《酒店达人》在内的大部分应用都不是特别方便。那时我的第一反应和同他开玩笑说手机买大了，但仔细想想，有什么样的改进可以帮助这样的用户？

巧合的是，我在当时见到了科大讯飞的产品。设

语音搜索功能和关联服务推荐

想一下：一个出门在外的商务人士，一只手拖着拉杆箱，一只手握着手机，此时如果允许他进行语音输入，是不是很酷?

小团队的优点就是执行力强，说做就做。于是业界首款支持语音搜索查询酒店的应用出现了，用户只需说出想要查询的地点，便可将参考点设置在该位置——这对于识图能力不强的用户是个福音。虽然目前的接入方式还比较初级，但已让我们的那位合作伙伴感到很满意，同时也为《酒店达人》未来的扩展提供了足够的想象空间。

关联服务推荐

我在2011中国•移动开发者大会上有幸结识了《易到用车》的同仁，他们以一种轻量级的运作方式，为有用车需求的客户提供非常贴心周到的一对一租车服务。在交谈中我们发现彼此的用户也很可能是对方的潜在用户。从用户给我们的反馈也显示，很多用户愿意尝试这类服务，因此互相关联的服务与服务之间的衔接推荐应运而生了。

当用户在《酒店达人》上预订了一家异地酒店，在确认订单时，酒店达人会询问用户是否需要针对该酒店的接送机服务，从而为有需求的用户提供一个方便快捷的服务通道。未来这样的推荐服务会越来越精准。

未来计划

《酒店达人》一路走来虽谈不上成功，但也颇为幸运。回头来看，是在一个恰当的时间点推出了一个迎合用户需求的产品，早了便成先烈，晚了则失去先发优势。

到目前为止，《酒店达人》的发展进程是根据我们对用户的理解来规划和实施的。近期同《易到用车》这类合作伙伴的接入拓宽了我们的视野，新的想法和思路不断涌出，当然这些想法需要得到市场和用户的检验。为此我们已踏上新的征途，相信在不久的将来，我们的产品会带给用户新的惊喜。

刘张博

早前求学于不列颠哥伦比亚大学，2007年进入移动互联网，曾就职于不同类型的移动行业公司。2009年底创办移花互动，致力于为手机用户提供出行生活信息及消费决策服务。

责任编辑：陈博（chenbo@csdn.net）

开放平台，有所为有所不为

UC首席运营官朱顺炎专访

记者 / 董世晓

朱顺炎
UC首席运营官

在移动互联网时代，浏览器已不再是单纯的上网工具，而是重要的移动互联网入口，在用户的日常生活中扮演着越来越重要的角色。6月26日，UC正式发布了其游戏开放平台战略，全力推广浏览器上的HTML5游戏。近日，《程序员》记者就读者所关注的几个问题，采访了UC首席执行官朱顺炎。

《程序员》：目前HTML5游戏的表现力还为人所诟病，UC此时推这个方向，是怎样考虑的？

朱顺炎：技术是逐渐完善的，因此，我们判断，随着HTML5协议标准的日臻成熟，HTML5游戏会成为移动游戏行业爆发的机会。目前市面上的HTML5游戏还是休闲类的比较多，也有一些社交类的。

《程序员》：有的开放平台会提供开发工具，UC开放平台是基于浏览器的，是不是只用HTML5就可以了？还是会提供一些特别的工具？

朱顺炎：在我看来，开放平台之所以能吸引开发者，无非出于两点，要么是收益，要么是用户。因此，我们会着重在这上面花大力气。当然，我们也会根据自己的能力和市场的需要，推出一些辅助开发工具。但我们不会特意去提供自己专用的一些工具，除非那种标准组件，否则会严重影响用户体验。提供开发工具，并非我们的强项，我们奉行“有所为有所不为”，倾向于通过平台的能力去号召专业的开发商进行支持。

《程序员》：原来PC上的网页游戏，转到移动端HTML5游戏，会遇到哪些问题？

朱顺炎：网页游戏转成HTML5的应用运行在移动端上，按说是最顺畅、投资最低的一种模式。但一个不容忽视的现实是，移动端浏览器对HTML5的支持还不够完善，因此很多开发者在纠结到底用客户端的方式来解决，还是在开发过程中等待浏览器对HTML5的支持的成熟。我建议可以考虑三个策略。

其一，选择篇策略类的、交互和动画效果要求不是太高的游戏题材。

其二，用客户端技术实现网页游戏移植到手机。

其三，谨慎投入，边做边看，实现游戏与HTML5完善进度的基本匹配。

《程序员》：我们注意到UC游戏开放平台与开发者的分成比例采用的是五五，但与业界比较通行的三七和四六相比，看起来没有优势，当初为何做这样的决定？

朱顺炎：在做这个决定之前，我们也分析了市面上的各种分成政策。之所以最后还是这样做，在于我们坚信我们为开发者做得更多，并最终实现与开发者的共赢。

App Store、Google Play等市场类的分成与我们平台的分成政策是不能类比的，它们不会给开发者提供推广、服务器、运营等整体服务。在我们的平台里，五五分成指的是纯收入的分成，是除去推广费、服务器、带宽、运营成本之外的，对开发者来说是实实在在的收入分成。综合分析一下，你就会发现我们平台比那些表面上开发者占大头，实际上是还要付出很多额外费用的分成政策对开发者更为有利。

多角度着眼未来

《世界Online》策划总监罗维专访

记者 / 陈粲然

罗维在采访中提出了“未来产品”的概念，同时他认为在移动游戏产业，灵活、扩展性强的技术将成为主流，一些灵活性不够、扩展性不强，甚至跨平台支持不够的引擎以及框架容易被淘汰。

在目前的移动互联网格局下，游戏仍是用户付费意愿最高的应用类型。在iOS平台，早已有《二战风云》、《王者帝国》以及各类三国游戏等一系列取得了不错回报的产品。而在它们的背后，顽石、Tap4Fun、Triniti等公司也成了业界标杆。

从2012年起，Android平台赚钱难的印象也在开发者心中也有了改观，一系列Android平台的游戏取得了收入上的成功。除了《二战风云》这样原本就在iOS平台取得成功的游戏外，由本文的采访对象，广州谷得网络科技有限公司创始人、策划总监罗维亲手打造的《世界Online》是一款运行于Android和Symbian平台的手机网络游戏，它在2012年4月时在中国地区的月收入就已突破了1000万元人民币。

做“未来产品”

《程序员》：介绍一下自己和你们的团队吧。

罗维：我从1999年进入电脑游戏行业，一直从事游戏策划和媒体工作。2005年进入手机游戏行业，曾任职拉阔游戏策划总监。在手机游戏领域，我曾策划过《帝国OL》、《火焰VS》等多款手机网游。我在拉阔工作五年多，在《帝国OL》推出之后，我感觉自己已跨过了职业生涯的一个阶段。就像一个即将要离开家庭的年轻人，向往着独立，希望重新寻找创作的激情。

于是在2011年2月，我离开拉阔与伙伴一起创立广州谷得，专注于移动终端游戏开发与运营。创业初期的团队只有九个人，团队成员都是来自广东地区的业内高手。创业初期最大的挑战来自我们自身：如何跨越我们曾经创立的高峰，再做一个里程碑式的产品？我们认为如果做现有时代的产品，肯定难以成功，只能做未来产品，这对于创业团队来说，具有很大的风险。

《程序员》：您谈到了“现有时代的产品”和“未来产品”，能否请您解释下这二者的定义？

罗维：在大部分厂商的概念中，把PC游戏、网页游戏移植到手机上就是手机网游。

2009年，我在拉阔推出了《帝国OL》后，一大群的高仿品蜂拥而至，一年内诞生了百款各种回合制即时游戏，但生存率连3%都不到，因为它们在

做的就是“现有时代的产品”。而如今，《世界Online》出现后，似乎又碰上了第二轮的手机网游热潮，大量高品质的移植作品都出现了。现在无法评价这些产品最终会如何，但我认为它们无法成为里程碑式或者划时代的产品。

《世界Online》是谷得的第一款产品，我们开始做它时，Andoid在国内市场的占有率还不到5%，《世界Online》推出的那个月，Android的市场占有率也只有10%。而今天，Android的占有率超过40%。因此，从2011年来看，《世界Online》是“未来产品”，同期的还有《二战风云》、《神仙道》等非常出色的产品。这些产品随着平台的发展，引爆了新的浪潮。而在今天，如果还有以《世界Online》、《神仙道》、《二战风云》为目标制作的产品，那么它们就是“现有时代的产品”，哪怕它们超越了前辈，也很难再有突破性的发展了。目前谷得正在筹备第二款手机网游，这是一款和《世界Online》完全不同目标用户的产品，是一款轻度操作的武侠角色扮演游戏，我们希望找寻新的突破点。

《程序员》：做游戏时，最大的乐趣在哪儿？最痛苦的地方又是什么？

罗维：做游戏最大的乐趣就是得到玩家的认可，这是一种无可比拟的满足感。而最痛苦的地方，就是有时我们很难确认用户真正的需求，探索的过程非常辛苦。同时还有没日没夜的加班，从2011年4月~2011年10月，我们几乎没有休假，每天从早上9点工作到晚上11点。因为团队的核心成员都是股东，也都是年轻人，所以在创业的道路上，大家都拼了命。

灵活、扩展性强的技术将成为主流

《程序员》：《世界Online》所用的游戏引擎是你们自主开发的吗？

罗维：是的，服务器部分，我们主要使用自行开发的引擎，该引擎使用到Netty网络框架、Spring程序框架，再结合游戏相关特性自行开发了复用性强的功能组件，包括脚本系统、社交系统、充值系统等。各部分组合成为我们当前使用游戏服务器引擎，该引擎在不断的完善中，其方向是能兼容各种不同类型游戏的开发需求，简化开发难度，提高效率。

对罗维来说，最大的挑战在于跨越自己曾经创立的高峰

客户端部分，在任务事件、UI的编辑等模块上，多数使用自行研发定义的脚本系统，可以灵活、高效地开发游戏相关功能；另外还使用了动画编辑器、地图编辑器等工具来编辑游戏相关的功能，大大提升开发效率以及游戏展现效果。

《程序员》：您觉得未来哪些有引擎和框架会被淘汰？

罗维：市场推动着技术的不断前进，一些灵活性不够、扩展性不强，甚至跨平台支持不够的引擎以及框架容易被淘汰，例如J2ME。随着Android、iOS和HTML5市场的迅速崛起，能否灵活地支持、最大地发挥各平台的操作体验，节省各平台的移植时间，是未来引擎变革、更新换代的关键！

《程序员》：目前你们有用到哪些新技术？

罗维：目前我们除了有用各种编辑器在制作产品，还会不断研究以及改进一些技术来提升游戏各环节的制作效率以及展现效果。例如OPenGL 2D/3D、粒子效果、图片处理的各种算法以及对画

《世界OL》游戏界面

面的更大展现能力、更灵活的功能脚本系统、事件脚本系统、微端更新系统、跨平台框架等。在移动平台上，因为限制比较多，所以使用这类技术还是有相当难度的。

清晰定位目标用户是成功的前提

《程序员》：在您看来，优秀的游戏需要具备哪些特质？

罗维：优秀的游戏唯一需要具备的特质，就是要满足用户目前的需求，并且能够引导用户新的需求。我会通过亲自与不同的用户交谈来了解他们。我进入游戏行业十多年，一直坚持和我的产品玩家亲密接触。游戏里聊天、聊电话、聊邮件、聊QQ……除了工作上的朋友外，我所有的网友都是来自我游戏中的玩家。有些玩家甚至已交往超过十年，从他们学生时代开始到今天的成家立业。还有些玩家和我说：“初中玩你的游戏，现在大学毕业了，能不能加入你的公司？”这也许是一个很笨、很浪费时间的做法，但我认为，这比光看数据报表可靠。

《程序员》：《世界Online》目前的收益状况如何？在Java和Android平台上有哪些不同的策略？

罗维：2012年4月，《世界Online》在中国地区的收入突破1000万元人民币，收入来自我们自运营的版本和腾讯平台的版本。Java和Android平台收入各占50%。目前手机网游通常都使用游戏免费道具收费的模式。我想在未来一年内，这个模式不会有太大的变动。

《世界Online》是以Android平台为标准制作的产品，再向下移植到Java平台。在经营策略上，并没有太大的区分。两个平台用户的付费率差不多，Android用户的ARPU值相对高一些。想获得良好收益的前提是，必须能确认产品的目标用户群、有清晰定义、研究他们的需求、发掘新需求。

我们所瞄准的，是2011年下半年至2012年，移动终端类型在国内市场快速变化的时机，大量用户从Java平台转移到Android平台。而最早转移的用户是具有非常多移动终端游戏经验的用户。《世界Online》定位的，正是这些的年轻用户群体。

同时，《世界Online》的用户类型是以群体性用户居多。在中国，以群体形式进行游戏的用户，大多以军人、学生为主体，这也和他们本身的职业和生活有关，他们常年群体生活、群体娱乐，往往会很多人一起玩一个游戏。他们竞争性强，都是重度的游戏用户。

针对这一特性，我们在游戏中需要设计非常多的群体活动。例如“国家功能”，在这个功能内搭建非常多的团队管理、团队发展架构，主要用来促进用户的团体协作。

他们对游戏内容追求非常大，所以《世界Online》不断地更新。开服8个月，我们就推出了8个资料片，不断地推出新功能，以满足他们的需求。

《程序员》：目前你们开始进军iOS平台，对你们来说，这个平台有哪些挑战？

罗维：iOS平台主要是针对游戏质量，特别是在游戏画面上有更高的要求。同时，iOS平台本身有着相当大的制约。例如手机耗电、续航能力等问题，以《世界Online》的画面质量，iPhone实际只能进行约4~6小时的游戏。如果采取3D渲染，达到更精细的画面质量，对于需要频繁使用网络传输的网络游戏而言，部分游戏只能进行1~2小时。这对于移动终端游戏是一个很大的挑战。对于《世界Online》和谷得往后的新产品而言，我们需要从中寻找一个平衡点。

游戏模式与玩家需求契合度的分析

文 / 郑金条

本文将游戏开发者对游戏设定时遇到的瓶颈、常见的游戏模式与玩家的需求做了对照，探讨开发一款游戏时，如何从游戏类别、游戏功能、技术等多方面做好全方位的把控。

很多人都在思考：什么样的游戏能够在历经市场和用户的双重淘汰后还能被称为好游戏？什么样的游戏在发行多年后还能获得市场的交口称赞？

好游戏应该拥有三个层面的效能：单次体验的娱乐价值、游戏在延续层面的娱乐价值、玩家之间的互动娱乐价值。这些在显性层面上表现为：玩家重玩一款游戏的意愿、特定时间段内的访问频度、单次游戏的时长、对外分享游戏的意愿、在游戏中消费的意愿、对游戏公司品牌的认知度、参与游戏线上与线下延伸活动的意愿、参与游戏的反馈。

游戏设定时的限制性因素

而在现实环境下，游戏设定时所设想的内容更容易受两方面的影响。

一个是**开发者本身的视野和技术障碍**。开发者的视野和对游戏未来走向的态度基本决定了一款游戏的所有基础性设定。而诸如美术或UI呈现效果的折扣，或某些技术环节产生了实施瓶颈而导致部分游戏功能萎缩等技术瓶颈，往往将游戏本身设限在一个有边缘的框架内，使游戏的实施预期从理想层面现实化，再进一步萎缩为平庸化。

另一个是**对市场走向和玩家需求的预判**。特别是对市场走向的预判可能将直接导致游戏玩法、美术呈现效果和题材植入类型迎合了市面现成游戏的趋势，成为现有市场的后来瓜分者。尽管这样做能够在模式上降低用户需求不匹配的风险，但与此同时也可能随时面临着玩家喜好转嫁迁移的风险。开发者还需要考虑如何有效地在原先的游戏基础上架设出全新的游戏体验以完成用户从其他游戏的定向迁移。这个层面实现的挑战在于如何培养新的玩家接受并最终喜欢这一模式，更大的困难是如何将原先对某款特定游戏产生依恋的用户吸引，毕竟玩家做一个全新的尝试背后需考虑大量的成本因素。

显而易见，目前的游戏市场呈现出一个明显的特征：游戏的玩法模式和深度被玩家一眼看透（例如局限于数值的调整、道具和场景/副本的编撰更新）。而游戏的乐趣衍化为两个层面：第一个层面是耐心意识，以不间断的时间消耗和相对重复的任务属性驱动玩家的资源和能力累积以实现最终的越级提升（这是一种依靠时间投入的耐力游戏）；第二个层面是超人意识，直接在游戏中将玩家的特定需求（例如控制欲望或者独特欲望）最大化，要么以玩家之间直接的对照来获得精神满足，要么让玩家获得在游戏中的主宰权力（例如超级资源量或者无与伦比的装备属性），从而最终有引导性地将玩家归类，并明显地将资源倾向性地供应于最可能为游戏贡献效益的用户群体。

这些终究将成为未来游戏的核心模块吗？在Free-

To-Play全球风潮的驱动下（以App Store和Google Play为代表的App端免费模式化；以Facebook为代表的Web端游戏免费趋势），未来还能有全新的模式在游戏平衡、玩家娱乐以及开发者效益三个层面上做更好的协调吗？

游戏在功能层面的五种模式

事实上，大量游戏都是各种模式的相互渗透，特别是关卡进阶模式和资源进阶模式，我们可以略微在现有游戏中对不同类型的游戏从功能层面做些区分。

■ **资源进阶模式**。主要涉及资源累加、装备强化和玩家经验的提升，现在这几乎成为所有游戏类型中的主流趋势，包括诸如《Plants VS Zombies》、《神仙道》、《三国塔坊》、《二战风云》之类的MMOs、网页游戏和大量的社交游戏以及手机游戏。此类游戏的进阶乐趣更多在于宽泛资源的捕获（可支配货币系统、锻造强化系统、装扮系统、超级等级）和由此所带来的成就满足。

■ **关卡进阶模式**。相对典型的包括《Angry Birds》系列、《Gardens of Time》、《Temple Run》系列、《Amazing Alex》等。此类游戏的进阶乐趣则在于挖掘玩家在处理游戏解决方案方面的技能、判断力和协调性，在基本玩法不变的基础上，通过解决方案的差异设定和玩家对方案处理的极致追求来驱动玩家依靠现有技能一步步探索游戏。

■ **排列和循环模式**。主要依赖游戏简单易懂的核心机制、适度的变量、可循环的排列组合框架以及可接受的概率属性来驱动玩家长期乐此不疲地参与游戏。典型的包括剪刀石头布、捉迷藏、麻将、扑克、俄罗斯方块、斗地主或Tiny Wings之类的。此类游戏因为其循环属性和无限制的排列组合属性，再加概率要素使玩家在博弈和对抗娱乐中充满了各种未知的可能性，从而让游戏的用户基数具有不同人群的强拓展性和不同年龄辈分的强延续性。

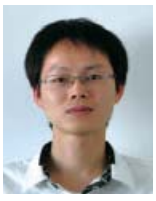
■ **UGC（User Generated Content，用户产生内容）模式**。主要为开发者在游戏中搭设交互框架由用户和用户本身产生游戏交互内容。目前依靠这一模式取得相对成功的社交（手机）游戏包括Zynga旗下的《Draw Something》（高峰期DAU超过2000万）、Fresh Planet旗下的《Song Pop》（目前的DAU超过220万）和On5旗下的《Charadium 2》。这些游戏的娱乐属性完全依赖于参与者的设定，而开发者只是提供了两者互为娱乐需求的平台，最终的娱乐效果能否达到预定的需求都依赖于玩家本身。

■ **简单娱乐模式**。这种模式可能和关卡、进阶、用户设定无关，仅是为了让玩家在打开游戏时能会心一笑，诸如较恶趣味的小游戏《大便超人》（输入相关名字就能进行恶趣味攻击，这种攻击甚至能够以动画效果直接呈现而不需要玩家任何操作的行为，玩家只是体会了全程的恶趣味画面而获得发泄式的满足）。

突出游戏的核心体验

以上只是从纯粹的功能需求进行定位，但确实大部分的游戏都能够对号入座。我记得曾有位开发者在谈到自己游戏的Demo版时不无骄傲地宣称他们的游戏拥有市面上经典游戏该有的各种元素，但似乎对于一款游戏而言，更为重要的是向玩家展示它的核心体验，而不是嵌入各种热门元素或者炫耀技术。

我常在想，一款游戏需要什么样的娱乐性来支撑玩家的持续体验、付费意愿和分享需求，甚至类似于Tetris（俄罗斯方块）这样历久弥新，在任何人群中都能够获得游戏共鸣？什么类型的沉浸能够抵消时间的逆淘汰在无穷的组合和变幻中仍能够保持一定的新鲜感？是在原先模式（诸如上文提到的资源进阶、关卡进阶、循环模式、UGC模式和简单娱乐模式）的基础上通过技术革新和故事梳理再造经典，还是找到另辟蹊跷的路径探索出全新的游戏道路，或许慢慢就能够明朗了。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

从零开始学游戏编程
可视化编程游戏开发工具学习指南

文 / 王楠

本文作者是一位游戏设计师，文中他将结合自身实践分析“门外汉”学习编程的难点，并分享利用可视化编程游戏开发工具学习游戏编程的经验。

开发游戏可能是学习编程的理由中最吸引人的一条了。但如何从零开始入门，达到能够开发游戏的编程水平，是困扰无数勇敢少年们的传统难题。作为一名游戏设计师，我没有系统地学习过编程。从5年前开始，我有了自己从头完整开发游戏的念头，于是断断续续地看了很多书，试过了很多入门方法和开发环境，但直到近半年才找到正确的门路。现在我在Unity开发环境下独立制作游戏原型和利用成型的框架完善游戏功能已不成问题。

本文会介绍如何从零开始学习游戏开发编程的方法，希望能为和我一样挣扎在编程大门之外的游戏开发爱好者们提供帮助。不过事先要说明的是，这种学习思路是为了帮助你在做游戏的过程中逐渐学习编写程序，不适用于其他领域，但作为一种入门方法，它能让你在半年到一年的学习之后，做到独立制做小游戏（或原型）。

门外汉学编程的难点

在介绍学习方法之前，我们先看看门外汉学编程最常遇到的问题。

第一，程序员们经常说程序语言只是编程工具，但市面上常见的教程都喜欢从语法、算法和程序语言的使用思想开始教学，而不是把编程语言当做解决实际问题的工具来入手。因此，初学者经常耗费很大精力才能理解书上写的算法和思想，却完全不知道理解之后能用来做什么。

第二，很多编程教程虽然配有实例，但一方面例子的学习难度曲线增加得很快，刚看完一个“Hello World”实例，下一个例子可能就变成教你如何分配内存（真实的故事，我的一本学习Objective-C的教程就是这样的）。另一方面初学者在对开发流程不熟悉的情况下，很难做到举一反三，从一个实例里总结出做另外三个游戏的方法，我经常遇见看了三个不同类型的游戏实例，放下书后却连一个游戏都做不出来的情况。

图1 从Hello World到实际的游戏项目之间，有一条门外汉难以跨越的鸿沟

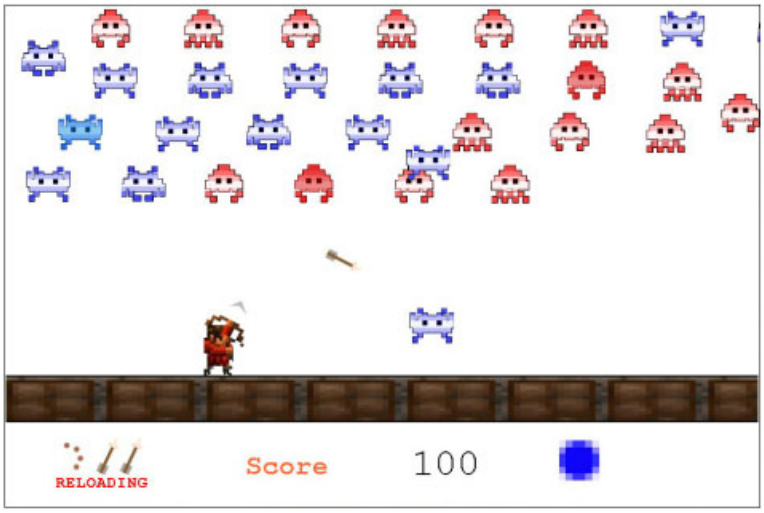


图2　在Construct 2下用了7个小时制作的游戏原型，包括一个特别的同色连击系统

第三，很多编程教程为了提高普适性，在使用现成架构方面都很保守，导致了很多重复造轮子的教程出现。例如在前几年Cocos2D（一个用于iOS平台游戏开发的游戏引擎）还没有现在这么火时，几乎所有的iOS游戏开发教程讲的都是如何使用OpenGLES来制作游戏图像，而这些底层架构的实现对初学者来说是根本不可能完成的任务。

因此，适合初学者的学习方针是：从实际需求出发；“怎么做”优于“为什么”（为什么可以在入门之后再慢慢理解）；使用允许你偷懒的工具或架构（需要做的越少越好）。这些要求其实很容易满足，答案恰恰在看起来和编程关系不大的领域——可视化编程工具里（Visual Programming Tools）。

可视化编程游戏引擎让你先做再想

可视化编程泛指一切使用可视化元素的操作代替文本输入的程序设计方式，大体上就是像画流程图一样通过连接若干“盒子”和“箭头”来实现程序逻辑。这个概念在游戏开发工具上的应用越来越流行，近年来还有井喷趋势，从老牌的GameMaker、RPGMaker、TorqueGameBuilder、到新兴的GameSalad、Construct 2和Unity都是其中的代表。尽管这些工具和引擎各有不同的开发方式，但它们都能让初学者在完全不懂编程语法和复杂算法的情况下快速实现自己的游戏设计。

我之前的态度是宁可抱着“看也看不懂，看懂了也不会做”的书苦学XNA（一个微软发布的使用C#的游戏开发架构）和Cocoa（苹果发布的使用Objective-C的应用开发架构），也不屑于使用GameMaker、GameSalad之类的图形界面开发工具。认为这些工具属于“业余型”，就算能做出游戏来也是旁门左道，不能修炼内功。

直到有一次参加了柏林独立游戏BIG Jam的活动，接触了很多非常优秀的游戏开发者。他们大部分人都把GameMaker和Flash这些简单的工具当做制作独立游戏的最佳选择。原因是他们多年以前开始学习游戏开发时使用的就是这些工具，常年的使用经验让他们能在最短的时间里用这些工具实现想法。而使用这些工具从头到尾制作了大量游戏的经历，也在他们以后学习用编程语言开发游戏时打下了很好的基础。

从那之后，为了快速开发原型，我开始物色入门级的可视化编程游戏引擎。HTML5游戏引擎Construct 2偶然进入了我的视线。花十几分钟学习教程实例之后，我很快用几个小时做出了一个一直在构思的游戏想法（当然想法本身就很简单，而且制作过程中碰到实现困难的设计都进行了进一步简化）。说来惭愧，尽管在主机游戏业从业多年，这次使用Construct 2的开发过程中我第一次感觉到对游戏开发的整个过程和架构有了初步认识。

首先，可视化编程工具里一般都有一个现成的游戏场景（任何游戏开发过程中都需要一个画布或一个摄像机来描述玩家可以看到的图像范围），然后你需要把游戏中需要的各个元素（一般称为Actor，例如主角、敌人、子弹等，这就是编程语言里对象的概念）放进场景里，然后通过关联逻辑模块来让它们快速互动起来。Construct 2的逻辑模块使用了非常贴近编程语言的按行号从上到下的执行顺序。而且你将从教程中学习到，原来游戏开始运行后每一帧都会按顺序执行一遍所有的逻辑，这就是游戏开发的基本框架中最常说的主游戏循环（Main Game Loop）。

除此之外，用户使用逻辑模块时不用担心语法错误和算法的设计，一般这类引擎里都会提供大量现成的算法模块可供挑选。只要专注于设计游戏

逻辑，其他事情可以说都是软件自动帮你完成。在观看教程和其他范例项目时也一目了然，学习别人的设计思想更加容易。

通过使用Construct 2独立完成了第一个游戏原型后，我学到了相似的游戏元素可以共享一部分属性（编程语言里使用类和继承的概念）；学到了所有活动的游戏元素都需要在每一帧的循环里进行驱动，每帧只运动一小段距离；还学到了应该在主游戏循环的什么位置判断是否Game Over，以及Game Over时进入另一个循环来等待玩家重新开始游戏等内容。

这段经历让我认识到有能力从头到尾制作游戏（或者原型）对于游戏开发的理解有多么重要。但有一个问题出现了——如果可视化工具那么好用，那为什么还要继续学习编程呢？我当时也光顾着高兴了，并没有从可视化编程工具转到真正编写代码的计划，直到……

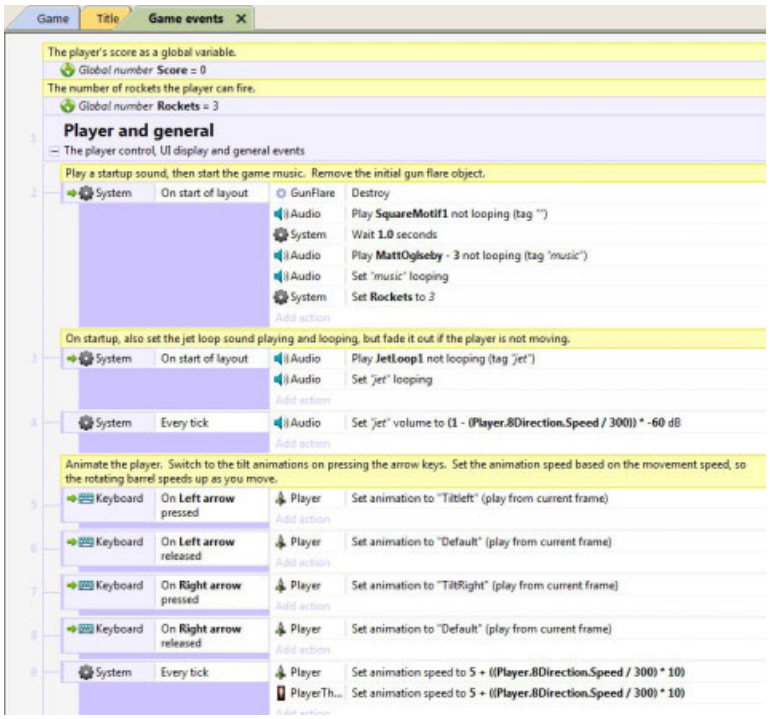


图3　Construct 2的Event编辑器其实就是主游戏循环的逻辑描述

由需求出发向编写代码的转型

直到我打算做个稍大一点的战略游戏项目，才开始在各种游戏开发工具中碰壁。接连尝试了Construct 2、GameMaker、Stencyl，可不管哪一个工具都无法很容易地提供我所需要的数据结构。重新审视了需求的增加和工具的局限性之后，我才决定开始学习Unity下的C#编程。在请教了团队里的程序员和有针对性地学习了一部分数据结构知识后，我终于在Unity中搭建出了设计需要的基本游戏结构。并在之后开始正式学习Unity的C#脚本，一步步地掌握了C#里类的继承、列表和字典的使用与委托等难以读懂，学了也不知道怎么用的概念。

刚开始在Unity开发环境里独立制作游戏原型时，我也感觉从零开始独立完成所有的代码非常和困难。但幸运的是，Unity引擎有个特点，就是有一个内容非常丰富的插件市场，其中最流行的插件类型之一就是可视化编程插件。这些插件（PlayMaker、uScript、Antares Universe等）将Unity游戏开发的常用功能打包成函数块，并按需求类别归纳成组，只要花一点时间阅读手册就相当于掌握了Unity里大部分的常用函数功能。之后用户通过连接不同函数块的输入输出接口来实现完整的游戏逻辑。

当时我为了开发一个动画状态很多的动作游戏原型，购买了PlayMaker插件，并且使用这个插件强大的状态和动画控制功能第一次学习了有限状态机的制作和使用。之后又经身边的程序员指点了解到有限状态机在游戏开发中的运用非常广泛，不光是控制动画，还可以用来控制菜单和不同条件下需要不同处理方式的很多游戏逻辑。

就像之前的Construct 2一样，我也只在一个项目

图4　使用PlayMaker用流程图的方式设计状态机，再填入相应的功能

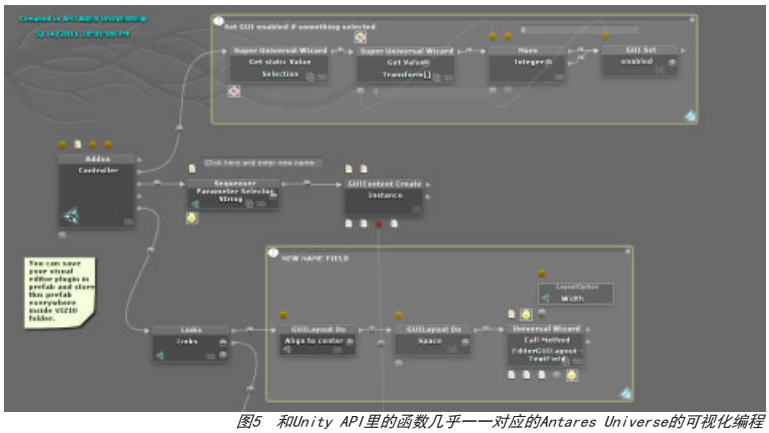


图5　和Unity API里的函数几乎一一对应的Antares Universe的可视化编程

中使用了PlayMaker的状态机。在了解了有限状态机的基本工作方式后，我在之后的项目中都使用了团队里的程序员开发的纯代码的有限状态机系统（开源的exUnity，还包括其他Unity下的游戏开发实用架构：https://github.com/jwu/exUnity），完成了从可视化编程到代码编程的转型。

这就是Unity下的可视化编程插件很适合用来学习编程的原因：和其他较为简单的工具的区别在于，Unity使用JavaScript和C#作为脚本语言，这个环境下的可视化编程插件只是把C#函数和脚本打包成了可视化的逻辑块，并没有改变其设计思路。

对于初学者来说，用可视化插件组装起来的游戏逻辑和用C#手动编写的游戏程序几乎是一一对应的，有时甚至能精确到函数段落（例如Antares Universe里的函数块就和Unity的全部函数功能一一对应）。在已熟悉整个设计流程的情况下，只需查阅Unity官方的脚本参考手册，就能完成从可视化编程到文本编程的翻译。我经过可视化工具的启发，很快就发现插件有些臃肿和烦琐，也无法实现一些需求实现。在接下来的两个原型中就越来越多地开始手动编写代码，对工具的依赖越来越小，直到完全抛弃。

现在可以回答前面提出的“为什么有了不用编程就能开发游戏的工具，还要学习编程”的问题了。如果你的所有设计需求都可以被可视化游戏开发工具完成，那么确实不需要进一步学习编程。但如果有的需求无论如何都不能用其他工具完成，那么自己写代码来实现就是唯一的出路，这时你有强烈的需求和目标，就可以通过询问或搜索“怎么做”来学习编程并满足需求。我的程序员好友经常说：“能否学会编程其实只取决于你的需求是否强烈，不得其门而入、或半途而废的都是需求不够明确或不够强烈的人。”

可视化编程工具对于游戏开发者来说就是一个筛选需求的过程：在硬啃编程书籍时，感觉自己有100个需求，但都不知道从哪开始学习、如何去实现；使用可视化工具，可以轻松实现90个需求，剩下10个就被放大并明确化了。接下来依靠上网学习或向他人请教，终究也能自己实现（个别超出能力的需求不要强求，请别人做或者放弃都比钻牛角尖要好）。

可视化编程工具能够培养我们由浅入深的思考习惯。先尽可能地用简单的逻辑去实现设计，如果用“盒子”和“箭头”无法完成，那么你在寻求代码上的解决方案时，就回答了“为什么要用这样的程序设计思想”的问题。经历了这个过程，你对“为什么”的理解会比一开始就去看专门讲解“为什么”的大部头程序书籍深刻许多。而通过实践理解了需求和程序设计之间的关系后，再去系统地阅读程序设计教程效果会好得多。

可视化编程虽然依赖于工具，但也能帮助你时刻把“程序语言即工具”的概念装在脑袋里。之后无论换什么引擎，用哪种语言，首先应该问自己：“这个工具能帮我做什么？我要怎样做才能实现需求？”另外，程序设计和画流程图之间的距离没有想象中那么大，通过反复用可视化编程工具画“流程图”的过程，能够从实践中学习各种游戏设计的实现方法，当你能准确地画出逻辑完美的流程图时，离你写出同样逻辑完美的程序距离已不远了。更重要的是，在这个过程中你实现了自己的想法，创造出了和书上的例程完全不同的东西，对于增强信心和进一步明确自己的学习需求的作用都是巨大的。

想学写程序，就要做程序员的朋友

最后讲下如何从身边的程序员那里获得帮助。初学者想要学写代码，有个程序员朋友能让你获益良多。当你遇到难题时，请把询问的重点集中在需求思路和关键字上，而不是一味求代码。高手

提供了思路以后自己实现，或通过关键字自己寻求解决方案，都更有助于水平的提高。

从可视化编程到代码的转化中，也可以尝试使用程序员写的功能库或常用的架构。前文中提到不要重复造轮子，当抛弃可视化编程工具时，就要用现成的功能库来代替。在选用功能库时，自己信任的程序员推荐的东西总会是一个非常不错的选择，他能告诉你一个库的优缺点并且在使用过程中提供技术支持。

我在学写代码的过程中，先是自己用最简单的方式实现功能，然后一边不断阅读和学习同个独立开发团队里程序员的项目结构和代码，再使用程序员设计或惯用的架构来组织自己的代码，这样既能最快地完成工作，又能逐渐养成较好的编程习惯和深入理解程序设计思想。

附录：流行的可视化编程游戏开发工具及简单点评

GameMaker

GameMaker是已有十多年历史的老牌独立游戏开发引擎，也是在世界范围内最受独立游戏开发者欢迎的引擎。巨大的用户基数和独立游戏圈用户们乐于分享的精神使得学习GameMaker非常容易。该引擎有自定义的脚本语言GML，方便用户使用脚本代码实现更高级的功能。最新版本的GameMaker Studio可以发布到iOS、Android和HTML5等各种平台。

官网：http://www.yoyogames.com/make

Construct 2

比起GameMaker学习起来更容易的HTML5游戏引擎，它的事件编辑界面（Event Editor）非常接近写单一过程的程序代码的模式和习惯，可以帮助初学者学习最基本的游戏循环构架。Construct 2本身的泛用性不如GameMaker，但仍然处在高速发展期，新功能添加的很快。

官网：http://www.scirra.com/construct2

GameSalad

这是一个很受业余爱好者和游戏开发初学者的图形界面引擎，逻辑功能全部都放在一个Library里，用户只要把需要的功能拖拽出来，填上参数，再和其他功能建立连接就可以实现游戏逻辑。只能在Mac下使用，可以发布到iOS和Android。

官网：http://gamesalad.com

Stencyl

使用Flash内核的游戏开发引擎，总的来说和GameSalad比较接近，可视化编程的部分由很多拼图积木组成。逻辑积木的组合方式比较灵活，可以尝试很多解决问题的思路。

官网：http://www.stencyl.com

Unity

非常流行的民用商用二合一引擎，不过其可视化编程的模块（PlayMaker、uScript、Antares Universe）需要另外购买，对于初学者来说，最大的好处是一开始可以把大部分的功能交给工具完成，然后一点点地添加必要的代码，直到可以完全不借助工具也能写出完整的游戏原型。

官网：http://unity3d.com

王楠

毕业于浙江大学，2008年加入德国YAGER工作室至今。2010年联合组建了aBit Games独立游戏开发团队，首个作品《Super Sheep Tap》获得IGF China 2011最佳音效奖。

责任编辑：陈博（chenbo@csdn.net）

结合场景的HBase性能分析

文 / 邓明鉴

在实际应用中，有很多细节能极大地影响HBase的性能。本文结合具体的应用场景对HBase的性能进行分析，指出了影响其性能的具体细节和解决方法。

HBase是一个分布式、可扩展的基于HDFS的大数据存储产品，可用于拥有海量数据的在线服务。目前，Facebook、Adobe、eBay、Yahoo!和Twitter等国外大公司都在使用它。国内起步相对较晚，因为不少人在做系统选型时对性能比较看重。我认为作为一个平台级产品，性能确实很重要，但它并不是一两个简单的Benchmark可以说清楚的，结合具体应用场景的分析更贴近实际应用。

关于Benchmar

NoSQL产品吸引人的地方通常是它们的Benchmark，因为它们的单机TPS可以远远超过关系型数据库，而且很多NoSQL产品都能够做水平扩展。但我们在追求高TPS的同时也容易走入一些误区，其中最重要的误区是没有分清Benchmark所针对的应用场景。虽然很多产品的Benchmark数据看上去很高，但一到实际应用场景中性能就上不去或者不能满足需求。造成这一现象的原因就是实际应用中需要针对应用来设计不同的参数，而不同的参数对性能的影响大不相同。另外，不同的数据对性能的影响也需要考虑。

一般来说，实际应用场景通常是在项目开发时由PM或PD评估出来的，包括以下这些情况。

- 有多少热点数据？缓存命中率大约是多少。
- 读写比例如何。
- 单条数据大小大约是多少，有多少列。
- 数据需要保存多久，总体数据有多大。
- 随机访问还是顺序访问。
- 单连接还是多连接，批量请求还是单次请求。
- 是否需要跨机房访问。
- 是否会出现瓶颈在客户端的情况。
- 索引大小。
- 数据是否需要打开WAL。
- 对响应时间的要求。

以上是一些常见的场景，还有一些是针对产品的特殊场景，主要是依据存储产品的自身原理来总结。比如，对HBase来说，数据是用3个备份因子还是2个，compact和split是否可以关闭，等等。

图1展示了一些我们用过的性能测试场景，供各位参考。

小结：在具体应用场景中做测试，得到的测试结果通常和理论值差距非常大，所以产品标明的Benchmark通常只能做一个参考，我们必须详细分析它为什么能达到Benchmark的数据，然后再结合自身的应用做进一步的判断。

吞吐量和响应时间

对在线产品来说，衡量性能通常体现在两方面：足够高的吞吐量和足够短的响应延时。足够高的

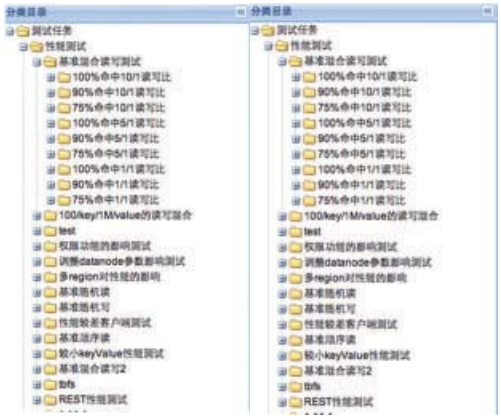


图1　性能测试场景

表1　性能数据

k/v=50byte/100byte随机读	1cf1q	1cf1q	1cf1q	1cf1q
client线程数	20	50	100	200
Qps	34606	42446	50372	64703
Response time(ms)	0.57	1.27	1.98	3.09
datablockcache实际命中率	100%	100%	100%	100%

吞吐量意味着系统能承受更高的并发，足够短的响应延时则意味着单次响应时间足够短。两者通常需要一个折中。比如，我们在对HBase 0.90做读压力测试时，得到了表1中的一组性能数据。

简单说明一下，1cf1q的意思是value体现为一个ColumnFamily，并且只有1个qualifier。这是最简单的情况。客户端使用5台机器，第二行标明了每个客户端使用多少线程并发读取。最后一行的datablockcache区别于HBase日志中的命中率，它是数据的命中率，排除掉了bloomfilter的命中率（实际上bloomfilter的命中率很高，会造成命中率看起来很高的假像）。

从表1可以看出，随着吞吐量的上升，响应时间也在增加，但并不是一个反比关系。对在线服务来说，响应时间通常有个最大接受值，所以可以先确定最大可接受的响应时间，然后在这个范围内尽可能多地提高吞吐量。

在吞吐量不大时，由于排除了排队、锁等因素，所以响应时间可以达到理论值。

对HBase来说，写响应时间通常在2ms以内。原因是每次写操作的顺序是先写WAL再写memstore，都成功则返回给客户端，由于memstore是本地的内存操作，响应时间在微秒级，可以忽略不计，因此时间主要消耗在客户端与服务器之间的RPC通信和写WAL上。客户端与服务器的RPC次数通常只有一个来回，除非需要重新location（location的RPC次数上限为3次，通常1次就够了）。而写WAL则是执行datanode的pipeline复制过程，顺序地往本机及另外两台datanode上写数据，再顺序ack回来。这个过程伴随着5次顺序的RPC调用。而在每台datanode上写数据则是一个顺序写的过程（hlog的append是顺序写），并且通过操作系统的pagecache来缓冲，因此写数据本身的时间是非常快的，接近内存操作的速度。因此，整个写响应的理论时间为6～9次RPC调用的时间，按0.1～0.2ms的RPC时间计算，写响应时间通常<2ms。

读响应时间则完全取决于命中率，如表1所示，内存命中率为100%时，响应时间最短可为0.57ms，如果是单线程会更快。内存完全命中时，响应时间就是1～3次RPC的请求时间。而内存不命中时，响应时间就取决于I/O次数了。从原理上讲，HBase一次读操作所需要的I/O如下。

- 读取每个storefile的checksum文件，判断文件是否完整。
- 读取每个storefile的bloomfilter，判断数据是否在该文件中。
- 如果bloomfilter为真，则读取该storefile的blockindex，来找到对应的block。
- 读取该block，并加载到blockcache中。

其中，blockindex只要读取一次就长驻内存不淘汰，因此大多数情况下I/O次数是2n+m次，n为storefile的数量，m为所请求的数据在几个storefile中（由于有版本的概念，通常所请求的数据很有可能同时在多个storefile中）。SATA盘的随机I/O时间大约为8ms，因此完全不命中时读响应时间大约在20ms左右。新版本的HBase在提高读性能方面做了很多优化，比如提高对元数据（bloomfilter）的Cache能力，把checksum这一次I/O想办法省去（读本地文件时不读checksum，或者把checksum信息写到hfile中）。此外，还有lazyseek（非常重要的一项优化，0.94以上版本）让读的storefile尽量少、compact优化算法等一系列优化都是为了减少n和m的值。

为了在实际应用中满足响应延时的问题，还需要对慢连接进行监控，也就是把每次响应的时间都统计下来，定期发给监控系统。这一点在0.92以上的版本中已得到支持。但为了取得更详细的数据，恐怕还需要根据自己的需求去修改代码。

至于吞吐量的性能分析，主要取决于排队等候的时间和锁粒度等。“hbase.regionserver.handler.count”是一个重要的参数，它决定服务器端最多启动多少个线程来同时处理RPC请求。一般设置100~200个线程可以极大提高吞吐量。排队的RPC长度限度为handler数量x100，可以通过系统监控来看实时的排队长度。如果排队长度始终为0，则说明服务器端还有能力并发处理更多的请求。很多时候吞吐量的瓶颈其实是客户端无法提供更多的并发请求了，此时可以尝试让客户端以多连接（注意区别于多线程）的方式请求服务器端（HBASE-2939）。

小结：在性能测试中要注意吞吐量和响应时间。一般情况下，我们应该先保证响应时间在一定范围内，然后再尽可能多地提高吞吐量。

热点数据、负载均衡及store和storefile的数量是影响HBase性能的几个关键因素。

几个关键因素

以下是影响HBase性能的几个关键因素。

■ 热点数据。热点数据即经常需要读取的数据。从上面的分析可以看出有没有命中内存，读性能的差距是两个以上的数量级，因此内存命中率对性能非常重要。

HBase自带blockcache，大小由“hfile.block.cache.size”参数决定。它能将每次读取到的数据以64KB为单位缓存起来，即一次随机读的最小单位是64KB，因此在设计上尽量将需要批量读取的数据前缀设置为相同或相近，这样可以极大减少随机I/O次数。当get命中blockcache时，会直接返回。假设每台服务器有10GB可用于blockcache，那么20台机器就能缓存200GB的数据。假设总数据为1TB，那么命中率为70%（50%+0.2/1）。这里的50%是blockindex，由于每次读取需要先读index再读data，而index长驻内存因此命中率接近100%。命中率可以通过regionserver的log文件来获取，或者通过ganglia来获得。

我们需要想办法尽可能多地提高内存命中率，比如减小rowkey大小，将版本数设置为1，设置合理的TTL。及时清理掉不必要的Cache，比如“hbase.rs.evictblocksonclose”参数（0.92以上版本）等。

■ 负载均衡。分布式环境中需要充分利用各台主机的性能，因此负载均衡是很重要的。当rowkey完全散开后，理论上每台服务器的TPS应该是平均的。HBase自带的balancer线程默认每5分钟工作一次，将region数平均分配到所有服务器上。但这并不是一个很好的选择，比如集群中有一大一小两张表，很有可能小表的region全部集中在一台机器上，但总的来说region是均衡的。这种情况下对小表的请求很有可能全部集中在一台机器上了。因此我们需要做表级别的balance。0.94.0版本已经有这样的功能了（HBASEHBase-3373），我们也可以很轻松地在之前的版本上实现。

实时监控请求分布是很重要的运维手段。如果负载不均衡，那么系统的性能测试是极为不准确的。因此需要把请求的分布实时统计下来，并定期发到监控系统展示。

■ store和storefile的数量。store是HBase的CF（ColumnFamily），每一个列簇是一个store。一般在设计的时候尽量不要使用过多的CF，因为当前版本的HBase在支持过多的CF时会有一些使用上的问题。比如，文件数会过多，多个CF在flush的一瞬间会有短暂的不一致（近期有些Patch在改进）；memstore过多等。一般尽量将列簇限制在3个以内，而尽量多地利用列。而且对同一行不同列簇的操作是串行的，因此性能上会有一定的影响。表2是我们针对多列簇的测试结果。

可以看出，列簇增加1倍，随机读的响应时间增加1倍，而列增加则对性能影响不大。

storefile是同一个列簇下的文件数目。文件数过多，会导致读响应变慢，这个前面已经分析过了，但好处是写会变快，原因是写入不受历史数据的影响。为了追求写入性能，可以关闭compact或放

慢compact的速度，但为了追求读的性能，需要适当加快compact的速度，比如使用多线程compact（0.92以上版本）。这个需要根据应用的读写情况进行适当的取舍。

另外，对于读取类应用来说，BloomFilter是必须设置的属性。如果列多的话，那么要尽量设置为“ROW”属性。如果是宽表（一行中有成百上千列），则可以考虑设置为“ROWCOL”属性。通常系统IOWait飙升都是由于BloomFilter没有设置造成的。

小结：我们需要尽可能多地提高数据的命中率，并根据命中率来推算需要的服务器数量。我们需要尽量保持服务器的负载均衡，并调整适当的balance策略。对读应用来说，要让storefile文件数尽量少，尽量使用列而不是列簇。

compact和split的影响

如前面所说，compact可以影响读写性能。split也一样，影响会更加严重。

- compact的影响：compact期间会将原有的storefile合并成一个新的storefile，这期间会增加网络和磁盘的I/O，因此会对系统整体性能造成一定的影响。如果把HDFS层加大是一个缓解的办法，比如一个HDFS集群上搭建多个HBase集群，那么compact的I/O冲击就平摊到了一个大的HDFS集群上，问题会相对较小。但如果compact采用默认的串行执行方法，则会在某些写压力很大的应用上堆积较长的compact队列。这会影响到hlog的滚动和回收，并且对读性能也有不利影响。所以建议对compact队列做监控，如果真的会有很长的队列，那么采用多线程compact会有很大帮助。

- split是在单个region过大时可以自动切分成两个region，这期间所有该region的请求会失败，所以我们尽量缩短split的时间或者减小split的次数。一个有效的办法是增大region的上限，让split次数减少，另一个有效的办法是建表时通过对应用数据的分析做出合适的pre-sharding（预分区）。

我们这里只分析第一种办法，因为这是系统层面的办法。增加region大小的参数是“hbase.hregion.max.filesize”，比如可以设成64GB。这样可以基本避免split发生，从而提高系统的写入稳定性。但增加这个值会带来以下几个问题。

表2 随机读性能测试

k/v=50b/100b随机	1cf1q	1cf2q	2cf2q	2cf2q
线程数	20	20	20	20
Qps（单台server）	9476	9514	4250	4702
Response time（ms）	2.36	2.1	4.7	4.25
CPU load	6.96	6.86	12.14	15.57
datablockcache命中率	60.29%	60.39%	60.9%	58.87%

- 首先是compact的压力会加大，因为要合并的文件更大了；
- 其次是由于compact变得更慢，导致要读的文件更多了，读响应下降；
- 最要命的是大文件使index变大，甚至到达几百MB或几GB，占用大量内存而且使加载速度变慢。

HBase 0.92以上的版本部分解决了这些问题。比如优化了compact逻辑，让compact去优先选择小文件合并，尽量减少大文件的重复合并，又比如使用HFileV2让index由平面结构变成树状结构，大大减小了index的大小。所以建议对split比较敏感的用户尽量使用0.92以上的版本。

小结：使用大HDFS集群可以有效降低compact的影响，但需要注意compact队列的堆积。split对系统稳定性的影响更大，建议通过pre-sharding以及增大region的方式来解决这个问题。采用增大region的方法尽量要采用0.92以上的版本，并注意读性能是否有下降。

datanode的选择

HBase底层的文件存储采用了HDFS，因此HDFS本身的性能也可以影响HBase。

2011年的Hadoop World大会上toddTodd Lipcon有一场经典的topic精彩的演讲，讲他们如何优化HDFS。另外，Facebook也提到过他们如何优化HDFS让它更适合做实时应用。总结起来看，HDFS在1.0以及Cloudera 3u3以上的版本中性能有明显的突破，主要体现在以下几点上。

- checksum大小可配置。可将原先的512字节做一次checksum可配置到64KB做一次，并且采用了优化的CRC32算法，让CPU的消耗大大降低。

■ 用Socket缓存。通过Keepalive机制来复用连接，减少大量重复新建连接的开销，并且重写了BlockReader，消除了不少数据拷贝。这一项我们测试随机读有13%、随机写有3%的性能提高。

■ 增加了本地读模式。如果判断到数据在本机，则直接跳过与datanode的通信，用直接调用文件系统接口来访问数据，并且由于数据在本机，可以忽略checksum这一步。这样一来随机读的I/O次数直接减少一半，实际测试结果也证明了这一点：仅这一项优化我们测试随机读性能可以增加100%以上。

另外，例如出错时重试机制的调整、Lease的及时释放等改进，都是针对实时应用所做的调整。自2011年以来，HDFS层面的优化大大提高了HBase的性能。由于Hadoop的版本分支非常多，目前性能有大的优化并且稳定的版本还是Cloudera 3u3以上，以及社区的1.0（0.20.205）以上版本，可以参考选择。

小结：不同的HDFS版本对HBase的性能影响也是不同的，需要多关注。

GC的影响

之前Hypertable的作者曾经认为，在HBase与Hypertable的性能对比中，让HBase落败的主要原因是GC。这其实是一个比较片面的观点。读完前面的内容，你应该比较清楚在实际应用中很多细节的影响其实要远远超过语言层面那一点点优化。当然，在理想状况下，GC的作用就会略为明显，但最多只有1%左右的影响。

在系统压力一般的情况下，比如8核CPU的load在3~4左右时，我们观察到的GC时间大约占0.5%左右，也就是说每1s会有5ms左右的Young GC暂停时间（2GB Young区），而CMS GC的频率并不频繁，影响也比Young GC更小。但在系统上线前最好多关注一下JVM参数，尽量避免Young区频繁GC，以及避免发生Full GC。在多列簇的情况下，比如2~3个列簇，由于碎片的增加，在很高的压力下发生CMS GC失败并引发Full GC是一件危险的事，有可能引发长时间的GC，这一点需要多关注GC日志。这个问题的解决方案可以是mslab分配内存方式，可以通过配置项来打开。但要记住在region数过多的场景下不可以使用mslab，它有可能引起OOM。

对于大内存的管理，Java的GC会有一定影响，将来的G1有可能从根本上改善这一状况，但目前G1还不太稳定，如果面对32GB、48GB或更大的内存，除了调整JVM参数外，使用native来管理内存也是一条路径。

小结：GC对性能的影响只是理论上的，并不足以改变HBase的整体性能。但对于稳定的线上应用来说，需要密切关注和监控GC的情况，避免出现长时间GC的事故。

总结

HBase的性能分析应该结合具体的场景，并对测试结果进行仔细分析，以便为线上应用提供尽可能合适的参数。目前hbHBase版本变化非常快，有很多性能优化点正在进行中，总体来说0.92版本已经比较稳定，可以尝试在生产环境中使用。

邓明鉴

阿里集团核心系统部技术专家，现主导维护和改进淘宝版本HBase，负责HBase的线上部署、规划及运维支持，熟悉各版本HBase源码，并有丰富的线上应用经验。

责任编辑：杨爽（yangshuang@csdn.net）

CloudStack架构详解

文 / Alex Huang

CloudStack是提供云计算服务的完整解决方案。它整合多种硬件资源，为用户提供透明的云服务，负责将虚拟资源映射到具体的硬件资源，并在用户间提供有效的安全隔离。

尽管CloudStack提供云计算服务的完整解决方案，但建立一个安全、可靠、可扩展的云计算数据中心仍需要详尽的计划和正确的部署，以及管理员的定期维护。本文将讲述CloudStack的软件架构、硬件部署架构和部署样例。

CloudStack对硬件资源的描述

在CloudStack中，每个Hypervisor（下文用Hypervisor代指运行该Hypervisor的服务器）是创建虚拟机的独立运算单元，多个同类型的hypervisor可以编组共享一个或多个Primary Storage（见下文）。在CloudStack中，我们将多个Hypervisor组成的编组称为Cluster。Primary Storage用于存储虚拟机的磁盘卷，在同一个Cluster中的Hypervisor可以通过网络访问Primary Storage，故同一个Cluster中的虚拟机可以实现Live Migration（动态迁移）。基于Primary Storage，CloudStack可以向虚拟机提供HA（High Availability）服务。当虚拟机所运行的Hypervisor出现故障时，CloudStack将在同Cluster中的其他Hypervisor重新启动该虚拟机，保证用户所购买的服务持续运行。

在实际的部署中，Cluster往往被作为数据中心的最小逻辑单元。用何种类型的Hypervisor组成Cluster、Primary Storage的容量及Cluster中各个Hypervisor管理的CPU和内存容量，取决于Cluster中所运行的虚拟机数量和虚拟机对性能的要求。一个设计谨慎的Cluster，通常会预留一定的冗余（CPU、内存和Primary Storage）以保证某个Hypervisor出现故障时，其他Hypervisor有足够的冗余接管该Hypervisor上运行的虚拟机。

在CloudStack中，多个Cluster编组成1个Pod。在数据中心的部署中，Pod通常对应一个机架（Rack），同一个Pod中的机器连接到同一个电源及同一个L2交换机，共享同一个广播域（Broadcast Domain）。不同Pod中的机器应该连接到不同的电源和L2交换机，以防止当某个Pod的电源或交换机出现故障后，其他Pod不能正常工作。基于这样的特性，Pod在CloudStack中被作为硬件错误的隔离边界。在部署一个数据中心时，应该将服务器规划为多个独立的Pod。为保证CloudStack数据中心的可靠性，应该至少包含2个Pod。

Pod之间的网络通过路由器（L3 switch）连接，其他网络设备（如NetScaler、F5和SRX）也由此接入网络并被多个Pod共享。防火墙同样设置于L3网络中，以控制公共网络对数据中心的访问。由于Primary Storage只能被位于同一个Cluster中的Hypervisor访问，加之Primary Storage频繁的I/O操作对网络带宽要求较高，所以CloudStack引入了可被全数据中心共享的Secondary Storage。磁盘模板、ISO等资源存放在Secondary Storage中。

Cluster、Pod、Primary Storage和Secondary

Storage是CloudStack所管理数据中心的硬件资源。当这些硬件资源被有效地整合在一起后，CloudStack将它们所代表的数据中心称为Zone。CloudStack可以管理一个或多个Zone。

CloudStack的软件构成

CloudStack由多个软件模块构成。CloudStack管理程序负责管理整个业务流程，并提供API。CloudStack在管理虚拟机的同时，也使用虚拟机帮助管理整个数据中心。这些特殊的虚拟机分别是：Secondary Storage Virtual Machine（SSVM）、CloudStack Virtual Router（VR）和CloudStack Console Proxy Virtual Machine（CPVM）。

在数据中心里，绝大部分操作都会产生数据交换。例如虚拟机与公共网络之间的数据交换、虚拟机之间的数据交换、虚拟机与Primary Storage之间的磁盘I/O、Primary Storage和Secondary Storage之间拷贝模板（Template）、ISO等。在达到一定规模的数据中心里，CloudStack必须能够管理成千上万的硬件资源、百万级的虚拟资源，以及亿计的数据。为此，CloudStack利用前文提到的特殊虚拟机帮助建立数据传播路径，执行特定操作，它们的存在对于CloudStack至关重要。下面对着几种特殊的虚拟机进行简要介绍。

- Secondary Storage Virtual Machine（SSVM）主要用于管理磁盘模板、ISO、磁盘快照，以及负责在不同的数据中心之间拷贝数据。SSVM必须具有两块虚拟网卡：一块连接到公共网络上，称作公共网卡；一块连接到数据中心的私有网络上，称作私有网卡。SSVM之间的数据交换通过公共网卡进行，CloudStack管理程序本身不参与这个过程。SSVM在下载用户指定的模板、ISO或上传磁盘快照时，管理程序只负责发送控制命令，不参与数据交换。在通常情况下，一个数据中心建立后，CloudStack会为其启动一个SSVM，管理员还可以根据需要启动多个SSVM，以应对有大量数据并发交换的情况。

- Console Proxy Virtual Machine（CPVM）用于提供远程桌面功能，用户可以通过它访问运行在数据中心的虚拟机。得益于Hypervisor的帮助，即使虚拟机本身关闭了一切网络端口，用户仍然可以通过CloudStack UI访问虚拟机。与SSVM类似，CPVM同样拥有公共网卡和私有网卡，CloudStack在数据中心建立后默认启动一个CPVM，管理员可以按需在同一数据中心中启动多个CPVM。

- Virtual Router扮演着DHCP、Gateway和DNS等角色，同时可以向虚拟机提供防火墙、负载均衡和VPN等网络服务。为了提供高可靠性，CloudStack提供Redundant Virtual Router功能，当一个VR失效后，备用VR自动接管前者提供的服务。VR所提供的网络服务以插件的形式构成，用户可以通过CloudStack管理程序。按需启动或关闭某个网络服务。

上述3种特殊虚拟机是CloudStack通过可编程虚拟机管理数据中心的典型应用，它们不需要数据中心提供除服务器、交换机和网络存储等常用硬件之外的特殊设备。

CloudStac 的部署拓扑

图1是典型的通过CloudStack管理部署的数据中心结构图。黑色实线代表控制指令的流向，黑色虚线方框代表CloudStack描述的硬件资源单元（如Cluster、Pod），灰色服务器图标代表运行有Hypervisor的物理服务器，蓝色服务器图标代表CloudStack创建的特殊虚拟机（SSVM、CPVM和VR）。

API请求首先到达的是负载均衡器（Load Balancer），它将请求转发给由多个CloudStack管理程序实例组成的阵列，该阵列中的CloudStack管理程序共享同一个数据库，当任一个管理程序失效后，阵列中的其他管理程序将接管前者管理的硬件资源，以此提供高容错性。CloudStack管理程序响应API请求，执行相应的操作，例如启动虚拟机、建立数据传输路径、控制Hypervisor等。

服务器端软件架构

CloudStack管理服务器（Management Server）致力于解决在数据中心运行云计算服务所面临的复

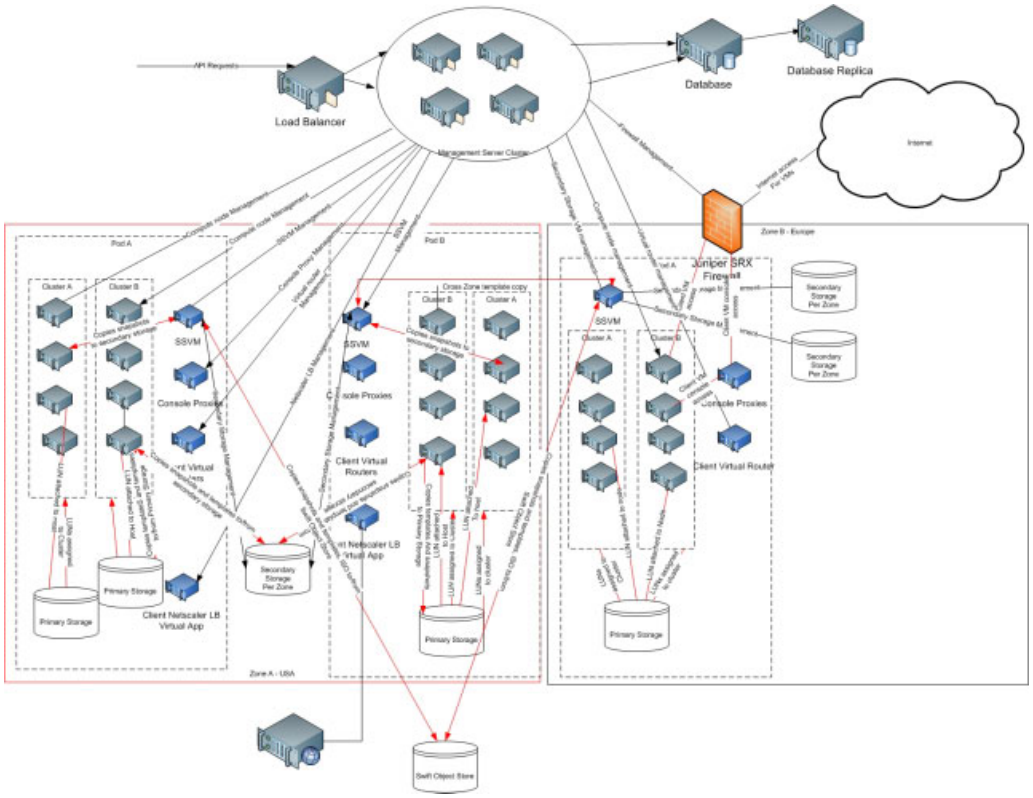


图1 通过CloudStack管理部署的数据中心结构图

杂性，可扩展性和可靠性等问题。

面向复杂的设计

CloudStack管理服务器必须处理好三类复杂性：控制（Control）、规约（Compliance）和选择（Choice）。云计算运营商想要控制在哪种硬件上部署云服务，控制用户的访问权限，以及控制系统中提供的云服务。同时，云计算运营商必须遵守所在公司或者政府的各种管理政策。为了能提供差异化的云服务，必须向终端用户提供可供选择的网络，磁盘和虚拟机等服务。所有的这些加在一起，使CloudStack管理服务器变得复杂。

CloudStack提供了插件（Plugin）机制，可以允许配置不同类型的硬件提供相同的服务。比如，Netscaler插件和F5插件都提供了负载平衡的服务。插件可以在加载CloudStack管理服务器的时候动态配置。

CloudStack管理服务器（如图2所示）的最上层是一个类似REST的API层。在API层，有一个可插拔的API引擎，可以允许在加载时动态添加其他类似REST的API。此外，还可以像在Tomcat中添加其他网络应用一样，添加那些非REST的API。CloudStack的图形界面就是API层的客户端之一。其他客户端，例如命令行界面（CLI），也可以在这个API层上进行开发。

一旦接收到请求，CloudStack会对调用者信息和调用请求进行一系列的安全检查。在这里可以集成已有的帐户和访问控制基础设施。目前，CloudStack只针对自己的数据库进行权限检查。如果这个请求只是一个简单的数据库访问，CloudStack会快速地读取数据之后返回。如果这个请求包含了复杂的云资源调配，比如部署一个虚拟机，CloudStack会先创建一个任务放到任务队列。CloudStack线程池中的另外一个线程会执行这个任务并将结果返回到任务队列中。客户端可以查询这个任务的执行状态。

当需要访问物理资源时，CloudStack管理服务器会发送命令给该资源的代理（Resource Agent），代理将转换命令并在该物理资源上执行，最后将结果返还给CloudStack管理服务器。

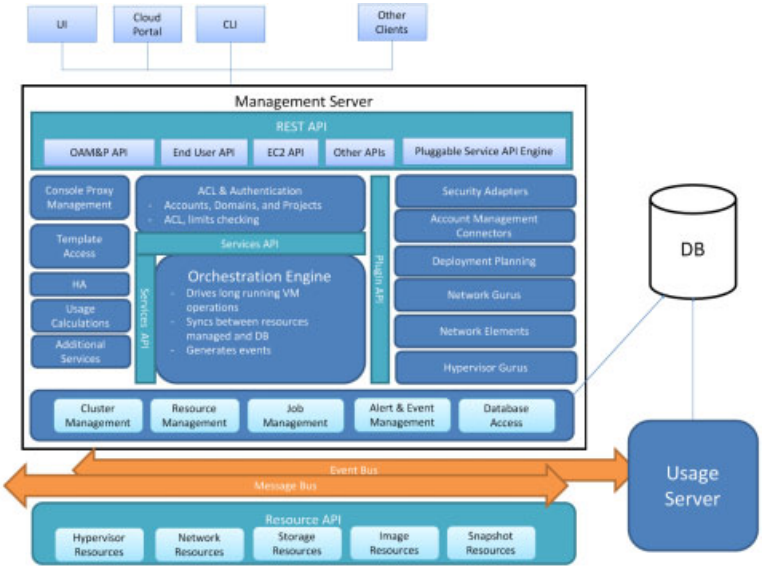


图2 CloudStack管理服务器

支撑整个CloudStack管理服务器的框架包含以下几个部分：集群、消息传送、任务管理、警报和事件以及数据库访问。

插件的架构

一旦一个任务被选出来执行，CloudStack的协调引擎（Orchestration Engine）就会全程参与协调调度执行。协调引擎并没有做具体的操作，它只是定义了完成一个云计算相关的操作所需要的一系列步骤。举例来说，CloudStack的协调引擎对部署一个虚拟机做了如下定义。

- 准备一个部署计划。
- 根据部署计划，准备虚拟机所需要的磁盘。
- 根据部署计划，准备这个虚拟机所需要的网卡。
- 启动这个虚拟机。
- 将虚拟机状态变化通知相关各方。

在以上每一步中，CloudStack的协调引擎调用插件进行具体的操作。例如，XenServer HypervisorGuru插件将生成用于发送到XenServer资源代理的StartCommand。

CloudStack的协调引擎用Java接口，清晰地定义了它所需要的各种功能。CloudStack的插件具体实现了这些接口。每个插件包含以下两个包：

- 第一个包是业务逻辑，部署在CloudStack管理服务器上。在这个包中，包含了对CloudStack协调引擎定义的插件接口的实现。如果它需要向管理员或者终端用户提供API，也可以把类似REST的API添加到API引擎中。
- 第二个包是服务资源（Server Resource），它可能需要和物理硬件放在一起，因为它不需要访问数据库。管理服务器和服务资源之间的通讯协议是JSON。

一旦插件被部署，需要更改components.xml才能使这个插件生效。CloudStack使用components.xml来动态配置需要部署的软件组件。更改之后，需要重启管理服务器。

表1是CloudStack支持的插件列表。

表1 CloudStack支持的插件列表

插件	功能
NetworkGuru	用于在物理网络上创建独立网络。VLAN是通用的技术。SDN是另外一种。
HypervisorGuru	用于处理各种虚拟监控程序对虚拟机的操作的不同之处。
NetworkElement	用于提供不同类型的网络服务，比如，DHCP，DNS，负载均衡，VPN，防火墙等。
DeploymentPlanner	用于提供在哪里创建虚拟机的各种算法。
SecurityChecker	用于检测帐号是否有权限访问虚拟实体。
HostAllocator	用于处理各种虚拟监控程序在计算虚拟机的容量上的不同之处。
StoragePoolAllocator	用于处理各种存储系统在计算存储容量上的不同之处。

面向可扩展和可靠性的设计

可扩展性和可靠性是CloudStack架构的一个关键设计。为了简化部署，CloudStack实际上将3个服务放在了一起。CloudStack正在把这三个服务分开，使他们可以独立扩展。这样可以对不同压力源进行针对性的扩展：增加输入请求并发度和增加可被管理的物理资源。

API服务集群运行在负载均衡器之后，接收用户的输入请求。它可以被独立地启动和关闭，而不会影响整个系统。它检查请求，判断是否需要协调服务（Orchestration Server）。如果不需要，直接执行这个请求，然后返回结果；如果需要，创建任务并放到任务队列中，供协调引擎使用。API服务同时也运行其他合作伙伴的服务。如果这些服务有问题，它们可以被安全地删除，而不会影响协调服务。

协调服务集群只用于处理云计算相关操作。如前所述，它定义了云计算操作所需要的步骤，使用插件实现相应的步骤。协调服务可以根据管理资

源的数量进行扩展。

资源代理被部署在每一个可用区域里。他们在CloudStack的命令和物理资源的API之间做转换。资源代理应该用物理资源支持的原生语言来实现。资源代理可以通过增加虚拟机的数量来进行扩展。

即使把这三个服务放在一起，一个运行在一般大小的虚拟机里面的CloudStack管理服务器和使用MySQL数据库就可以管理多达10000台左右的物理资源。同时，可以添加多个CloudStack管理服务器，组成一个集群，从而管理更多的物理资源。

总结

CloudStack的架构被设计成可以处理在真实世界中部署云计算所产生的复杂性、可扩展性和可靠性等问题。CloudStack是开放源代码的，而且已经被捐赠给了Apache基金会。欢迎大家访问www.cloudstack.org，并加入CloudStack的开发。你可以订阅我们的邮件列表：

- cloudstack-users-subscribe@incubator.apache.org（面向用户）；
- cloudstack-dev-subscribe@incubator.apache.org（面向开发者）。

Alex Huang

CloudStack创始工程师。曾担任AdInfuse首席架构师、OpenWave软件架构师。

责任编辑：杨爽（yangshuang@csdn.net）

把握本质规律

《数学之美》作者吴军专访

记者 / 卢鹄翔

无论是互联网，还是手机、电视，现代通信都遵循信息论的规律，整个信息论的基础都是数学。搜索引擎、语音识别、机器翻译也都是我们生活中离不开的技术，数学也是解决这些问题的最好工具。在《浪潮之巅》出版后，吴军将蕴含在这些技术中原本深奥难懂的数学知识和背后的故事，通过深入浅出的笔端，在《数学之美》中向读者娓娓道来。

《程序员》：《数学之美》集中阐述了对数学和信息处理这些专业学科的理解，可否再举一个数学方法彻底改变了计算机领域的例子？

吴军：在通信编码方面，有个称为Turbo Code的方法。以往电话传输语音时只能使用64Kbps带宽（PCM标准）。如果希望在其基础之上传输数据，也无法超越64Kbps的瓶颈，因此通过调制解调器上网最快只能到56Kbps。而利用Turbo Code则能将电话线扩展到很多频带。频带拓宽以后，再利用它提供的一套编码方法，能够实现容错、校验，保证数据平滑地传输。我们今天使用的DSL就得益于这项数学编码技术。

DSL技术在20世纪80年代前后就已经被提出，但直到90年代互联网兴起之后才真正得到普及和推广。因为在此之前，公司内部计算机往往使用同轴电缆相连，不需要考虑使用电话线传输大量数据。互联网兴起以后，家庭用户对电话线路传输数据有了迫切需要，为这项技术的实践提供了契机。Turbo Code还被用于计算机内部的通信，比如硬盘的控制器，它的应用使得硬盘的传输率大大提高，这样通信不会成为硬盘数据读写的瓶颈。今天的硬盘控制器都用到了这项技术。在Turbo

吴军认为，只有掌握算法精髓，在遇到需要的场合，才可能运用自如

Code的解码中，最重要的算法是BCJR算法，一个典型的应用数学的成果。这个算法，在《数学之美》书中也提及过。

《程序员》：统计学对提高自然语言处理效率发挥了重要的作用，是否还有其他一些数学知识对未来的计算机领域也将发挥重要的作用，值得程序员关注？

吴军：云计算兴起以后，大数据处理日益重要。大数据处理在很大程度上依赖于机器学习，因为对数据进行挖掘不可能由人完成。机器学习对各种数学工具的需求非常多。过去一些数学工具我们

没有看到特别的用处，比如线性代数，但今天很多分析都仰仗于此。比如PageRank、社交网络关系链分析都基于线性代数。

从前我们讲到机器智能，往往是针对一个小问题、小专家系统，这已经是公认的很难继续发展的领域了。然而在有了大数据之后，机器智能的问题被集中到了如何最快地从大量的、看似没有太多联系的数据中获取知识，这不仅是当今的热门话题，也是一个很有意义的事情。在这个过程中，以往研究人员觉得派不上大用场的数学工具又开始重新发挥重要作用。另外，最大熵模型中最优化理论的很多基础也是以往大家所忽视的，其中泛函分析就是一个比较纯粹的数学理论，但如今它也有很多实际应用。

《程序员》：对于大学生，以及工作后的程序员，如果他们希望进一步学习数学和算法的知识，你有哪些建议？

吴军：对于已经工作的人，在知道某个好方法之后，还要在做事的过程中有意识地使用这些好方法，这一点非常重要。

以中文分析为例，通常我们遇到的问题，可以使用常规语言模型得到比较好地解决，但某些特殊情况，比如诗词就涉及很多特定分词，而无法通过常规语言模型处理。此时有两个办法：可以通过编写一些特别的规则拼凑——这样写出的程序必然混乱；但假如你相信这样的问题不只会出现一次，也可以为此建立一个特定的语言模型，并结合常规模型一起解决。编程过程中涉及数学的情况非常多，关键看你用不用它——是相信存在一个数学模型能够解决，还是只图省事将问题绕过去，差别很大。

对于在校生，建议多在实验室参与项目实践，毕业以后在实验室之外也继续实践。我觉得国内学生学习课程时，课程设计做得太少，很多计算机系课程的编程量，可能连美国同类课程的1/10都不到。此外，如果阅读与数学相关的工程书籍，我建议选择从国外引进的译作，系统性和严谨性都好些。关于具体选择哪本，还需要看是解决哪个领域的问题。

关于算法，一直有“道”和“术”的说法，大部分专业书籍介绍的往往是具体算法，属于“术”的范畴，读者在阅读之后并不容易举一反三。与此类似，如果仅是了解一个数学工具或学会解决一个问题，也存在这样的弊端。我希望读者在阅读书籍之后能够进一步做深入的思考，真正掌握算法的精髓，在遇到需要使用的场合，才可能做到自如运用，这样才算从本质层面真正学会。否则就算了解再多算法，问题稍微变化，就会觉得无所适从。

产品是能够持续受到用户欢迎，还是昙花一现，很多时候就在于是否能用正确的方法处理问题，能否真正把握其中的本质规律。

《程序员》：在你的学习经历中，有没有希望与读者分享的经验？

吴军：我的学习过程应该与大家没有很大差别，但作为工程人员，我学习的内容可能相对多一些、广泛一些。我所幸的是，在成长的过程中，在学习和工作经历中，总能和很多各个方面一流的人共事，比如我在清华时遇到王作英教授、李星教授，在约翰·霍普金斯遇到贾里尼克院士、运筹学大师阿兰·高德曼院士，在Google遇见搜索名师阿米特·辛格院士、计算机系统专家R·凯茨院士等。我觉得一个人周围是什么朋友，是什么样的同事与自己能否取得进步的关系很大。在年轻的时候，选择工作时，宁可少挣些钱，也要寻找最好的成长机会。我所从事的计算机领域是一个应用广泛、机会很多的领域。但在这个领域中我也见到过许多年轻人，过早地考虑经济利益，放弃了很多学习的机会，以至于长期发展缺乏基础，运用所学知识也难以自如。因此，对自己是否有高要求，追求高的境界，好的方法，对能否取得进步的影响也很大。

《程序员》：你提到写作这本书也是希望IT公司的工程主管们能够带领自己的部门提高工程水平，对于工程主管，你有哪些经验希望与他们分享？

吴军：对于做工程来说，如果使用一个正确的好

方法，未必一定能够取得成功，因为在整个过程中还有很多非技术因素会发挥作用。但从长远来看，如果使用一个不好的方法和不正确的模型，几乎可以肯定这个产品不会取得成功。即使短期看起来有效，但做出的产品往往只是山寨的结果。产品是能够持续受到用户欢迎，还是昙花一现，很多时候就在于是否能用正确的方法处理问题，能否真正把握其中的本质规律。

《程序员》：Princeton大学出版社出版了一本《Nine Algorithms That Changed the Future》，其中也讲到许多关于搜索引擎、密码学方面的知识，可否谈谈算法和数学之间的关系？

吴军：算法很多时候是以数学为基础的，比如PageRank，我觉得影响更大的还有Viterbi算法，它是我们今天通信技术的基础。还有应用数学一些分支，比如运筹学、博弈论中的方法，在计算机科学，以及经济学上都有很多直接应用，这些方法背后的数学是许多计算机算法的基础。

《程序员》：你是国家核高基项目“新一代搜索引擎与浏览器”的总负责人，可否介绍一下这个项目和目前取得的进展。

吴军：这一项目是腾讯公司和清华大学联合申请的，因为要求以企业为主，因此就定了我做总负责人。这个项目的目的是构造一个全新的搜索引擎，不仅搜索质量要做到国内最好，此外还要结合社交网络，为用户提供个性化的体验。在浏览器方面，希望打造一个优化搜索体验的浏览器。我们知道，在社交网络中，可以搜索的内容比一般网页搜索要多很多。除了外网的内容，还包括用户自己的内容，用户好友的内容，社交网络中一般性，却是非登录态用户（包括网络爬虫）看不到的内容。如何将这些内容组织好，很具有挑战性。同时，在社交网络中的用户在登录时，我们可以通过他过去的习惯以及好友关系链等个性化的信息，提供比非登录态用户更加精准的结果。为了做到这些，需要打造适合搜索的云计算平台。此外无线搜索也会进一步完善个性化和本地化，因为无线终端是与个人绑定的，同时具有丰富的地理位置信息，这些信息不仅可以帮助把无线搜索做好，而且可以帮助商家的无线搜索上投放广告。在浏览器方法，很多搜索的功能都将以插件的形式直接放到浏览器中。

《程序员》：百度的搜索引擎已经做得足够好了，为什么腾讯还要花大力气，大投入做这件事？是否是眼红百度的广告收入？

吴军：2009年腾讯CTO熊（明华）总来硅谷请我加盟腾讯、帮助公司做好搜索时，我也问过他同样的问题。我说，这个题目对我来讲没有吸引力，因为不过是把这Google的工作重复一遍。对我来讲做一个和Google一样好的搜索引擎不难，难的是做出来以后，和百度的差异不足以使得用户需要切换过来。因此需要给我一个理由，包括用户能得到什么新的好处，否则即使做得再好用户也是不可能从百度切换过来的。我一直强调新的产品只有质变，革命性的变革才能战胜已经统治市场的产品。同时还需要他告诉我公司有什么特殊的资源推广，因为雅虎和微软投入了大量资源推广都不成功，国内的新浪、搜狐和网易也不成功。那么腾讯是否有资源，有决心做这件事，因为这是一件长期吃力未必讨好的事情。他介绍了腾讯的社交网络Qzone，这个和Facebook类似，里面的内容是外面没有的，同时登录台很多信息是可以利用的。有了这些信息就有希望比百度有质的提高。在推广资源上，腾讯在无线领域有得天独厚的优势，而现在恰恰处在无线搜索的爆发期。在决心上，腾讯将它看作五年后收入主要的成长点，放在战略高度。

虽然熊总第一次没有说服了我，但我们谈得还是很投缘，之后他带我和朱会灿到深圳见了公司的几位主要负责人，我们也对腾讯的业务和优势有了些了解，最终他们一起说服了我们。在我看来，搜索这件事，要么做成通吃市场，要么失败一无所获，几乎没有中间状态。当然，这些事情要做到，需要花很大的精力、很长的时间，不是照着Google做一个搜索引擎往QQ上一扔就完事这么简单。

SNS中的文本数据挖掘

文 / 顾森

在2012年6月刊中，作者给出了中文字符串能够成词的若干标准，从而实现了无知识库的自动抽词程序。在这篇文章中，作者将在抽词程序的帮助下，以词语为单位，在各个维度上挖掘人们在社交网络中的用词动向。

在大规模中文语料中，词语往往具有内部凝合固定、出现环境丰富等特点。我们可以借助这些特征，用数学方法刻画出一个字符串成词的概率，从而得到一种无需知识库就能从大规模语料中抽取词语的算法。对人人网的用户状态数据进行抽词，并与已有的现代汉语词库进行对比，我们便能得到一份新词列表，这将给中文分词等算法带来很大帮助。

社交网络中的热词现象

不过，我还想到了更有意思的玩法。为什么不拿每天用户状态里的词去和前一天的状态作对比，从而提取出这一天里特有的词呢？这样一来，我们就能从人人网的用户状态中提取出每日热点了！我手中的数据是人人网部分用户在2011年12月上半月发表的状态，其数据规模完全可以支持每日热点挖掘的工作。

表1　12个代表词

下雪	33	92
那些年	139	146
李宇春	1	4
看见	145	695
魔兽	23	20
高数	82	83
生日快乐	235	210
今天	1416	1562
北半球	2	18
脖子	23	69
悲伤	61	33
电磁炉	0	3

我选了12个比较具有代表性的词，并列出了它们在2011年12月13日的用户状态中出现的频数（左列的数字），以及在2011年12月14日的用户状态中出现的频数（右列的数字）。

大家可以从直觉上迅速判断出，哪些词可以算作是12月14日的热词。比方说，“下雪”一词在12月13日只出现了33次，在12月14日却出现了92次，后者是前者的2.8倍，这不大可能是巧合，初步判断一定是12月14日真的有什么地方下雪了。“那些年”在12月14日的频数确实比12月13日更多，但相差并不大，我们没有理由认为它是当日的一个热词。

一个问题摆在了我们面前：我们如何去量化一个词的“当日热度”？第一想法当然是简单地看一看每个词的当日频数和昨日频数之间的倍数关系，不过细想一下你就发现问题了：它不能解决样本过少带来的偶然性。12月14日“李宇春”一词的出现频数是12月13日的4倍，这超过了“下雪”一词的2.8倍，但我们更愿意相信“李宇春”的现象只是一个偶然。更麻烦的则是“电磁炉”一行，12月14日的频数是12月13日的无穷多倍，但显然我们也不能因此就认为“电磁炉”是12月14日最热的词。

忽略所有样本过少的词？这似乎也不太好，样本少的词也有可能真的是热词。比如“北半球”一词，虽然它在两天里的频数都很少，但这个9倍的关系确实不容忽视。事实上，人眼很容易看出哪些词真的是12月14日的热词：除了“下雪”以外，“看见”、“北半球”和“脖子”也应该是热词。你或许坚信后三个词异峰突起的背后一定有

什么原因（并且迫切地想知道这个原因究竟是什么），但却会果断地把“李宇春”和“电磁炉”这两个“异常”归结为偶然原因。你的直觉是对的——2011年12月14日发生了极其壮观的双子座流星雨，此乃北半球三大流星雨之一。白天网友们不断转发新闻，因而“北半球”一词热了起来；晚上网友们不断发消息说“看见了”、“又看见了”，“看见”一词的出现频数猛增；最后呢，仰望天空一晚上，脖子终于出毛病了，于是回家路上一个劲儿地发“脖子难受”。

数据的平滑

让计算机也能聪明地排除偶然因素，这是我们在数据挖掘过程中经常遇到的问题。我们经常需要对样本过少的项目进行“平滑”操作，以避免分母过小带来的奇点。这里，我采用的是一个非常容易理解的方法：一个词的样本太少，就给这个词的热度打了折扣。为了便于说明，我们选出4个词来进行分析。

表2截取了前四个词，右边四列分别表示各词在12月13日出现的频数，在12月14日出现的频数，在两天里一共出现的总频数，以及后一天的频数所占的比重。第三列数字是前两列数字之和，第四列数字则是第二列数字除以第三列数字的结果。最后一列应该是一个0到1之间的数，它表明对应的词有多大概率出现在了12月14日这一天。最后一列可以看作是各词的得分。可以看到，此时“下雪”的得分低于“李宇春”，这是我们不希望看到的结果。“李宇春”的样本太少，我们想以此为缘由把它的得分拖下去。

表2　4个典型词分析

下雪	33	92	125	0.736
那些年	139	146	285	0.512
李宇春	1	4	5	0.8
看见	145	695	840	0.827
（平均）			313.75	0.719

怎么做呢？我们把每个词的得分都和全局平均分取一个加权平均！首先计算出这四个词的平均总频数，为313.75；再计算出这四个词的平均得分，为0.719。接下来，我们假设已经有313.75个人预先给每个词都打了0.719分，换句话说每个词都已收到了313.75次评分，并且所有这313.75个评分都是0.719分。“下雪”这个词则还有额外的125个人评分，其中每个人都给了0.736分。因此，“下雪”一词的最终得分就是：

下雪 $(0.736\times125 + 0.719\times313.75) / (125 + 313.75)\approx0.724$

类似地，其他几个词的得分依次为：

那些年 $(0.512\times285 + 0.719\times313.75) / (285 + 313.75)\approx0.62$

李宇春 $(0.8\times5 + 0.719\times313.75) / (5 + 313.75)\approx0.7202$

看见 $(0.827\times840 + 0.719\times313.75) / (840 + 313.75)\approx0.798$

容易看出，样本越大的词，就越有能力把最终得分拉向自己本来的得分，样本太小的词，最终得分将会与全局平均分非常接近。经过这么一番调整，“下雪”一词的得分便高于“李宇春”了。在实际运用中，313.75这个数也可以由你自己来定，定得越高就表明你越在意样本过少带来的负面影响。这种与全局平均数取加权平均的思想叫做Bayesian average，从上面的若干式子里很容易看出，它实际上是最常见的平滑处理方法之一——分子分母都加上一个常数——的一种特殊形式。

实际结果

利用抽词程序抽取出人人网每天用户状态所含的词，把它们的频数都与前一天的作对比，再利用刚才的方法加以平滑，便能得出每一天的热词了。利用2011年12月10日到12月15日的数据，可以得出从12月11日到12月15日的热词（选取每日前5名，按得分从高到低排列）。

12-11：皇马、巴萨、卡卡、梅西、下半场
12-12：淘宝、阿内尔卡、双十二、申花、老师
12-13：南京、南京大屠杀、勿忘国耻、默哀、警报
12-14：流星雨、许愿、愿望、情人节、几颗
12-15：快船、保罗、巴萨、昨晚、龙门飞甲

看来，12月14日果然有流星雨发生。

注意，由于我们仅仅对比了相邻两天的状态，因而会产生个别实际上是由于工作日与休息日区别造成的“热词”，比如“教室”、“老师”、“星期

二”等。把这样的词当作热词可能并不妥当。结合上周同日的数据，或者干脆直接与之前完整一周的数据进行对比，或许可以部分地解决这一问题。

其他维度的数据挖掘

事实上，有了上述工具，我们可以任意比较两段不同文本中的用词特点。更有趣的是，人人网状态的大多数发布者都填写了性别和年龄等个人信息，我们为何不把状态重新分成男性和女性两组，或者“80后”和“90后”两组，挖掘出不同属性的人都爱说什么？要知道，在过去，这样的问题需要进行大规模语言统计调查才能回答。如今，在互联网海量用户生成内容的支持下，我们可以轻而易举地挖掘出答案来。

我真的做了这个工作（基于另一段日期内的数据）。男性爱说的词有：

兄弟、篮球、男篮、米兰、曼联、足球、蛋疼、皇马、比赛、国足、超级杯、球迷、中国……

下面则是女性爱说的词：

一起玩、蛋糕、加好友、老公、呜呜、姐姐、嘻嘻、老虎、讨厌、妈妈、呜呜呜、啦啦啦……

下面是90后用户爱用的词：

加好友、作业、各种、乖乖、蛋糕、来访、卧槽、通知书、麻将、聚会、补课、欢乐、刷屏……

下面则是80后用户爱用的词：

加班、培训、周末、工作、公司、各位、值班、砸蛋、上班、任务、公务员、工资、领导……

不仅如此，不少状态还带有地理位置信息，因而我们可以站在空间的维度对信息进行观察。这个地方的人都爱说些什么？爱说这个词的人都分布在哪里？借助这些包含地理位置的签到信息，我们也能挖掘出很多有意思的结果来。例如，对北京用户的签到信息进行抽词，然后对于每一个抽出来的词，筛选出所有包含该词的签到信息并按地理坐标的位置聚类，这样我们便能找出那些地理分布最集中的词。结果非常有趣：“考试”一词集中分布在海淀众高校区，“天津”一词集中出现在北京南站，“逛街”一词则全都在西单附近扎堆。北京首都国际机场也是一个非常特别的地点，“北京”、“登机”、“终于”、“再见”等词在这里出现的密度极高。

从全国范围来看，不同区域的人也有明显的用词区别。我们可以将全国地图划分成网格，统计出所有签到信息在各个小格内出现的频数，作为标准分布；然后对于每一个抽出来的词，统计出包含该词的签到信息在各个小格内出现的频数，并与标准分布进行对比（可以采用余弦距离等公式），从而找出那些分布最反常的词。程序运行后发现，这样的词还真不少。一些明显具有南北差异的词，分布就会与整个背景相差甚远。例如，在节假日时，“滑雪”一词主要在北方出现，“登山”一词则主要在南方出现。地方特色也是造成词语分布差异的一大原因，例如“三里屯”一词几乎只在北京出现，“热干面”一词集中出现在武汉地区，“地铁”一词明显只有个别城市有所涉及。这种由当地人的用词特征反映出来的真实的地方特色，很可能是许多旅游爱好者梦寐以求的信息。另外，方言也会导致用词分布差异，例如“咋这么”主要分布在北方地区，“搞不懂”主要分布在南方城市，“伐”则非常集中地出现在上海地区。当数据规模足够大时，或许我们能通过计算的方法，自动对中国的方言区进行划分。

其实，不仅仅是发布时间、用户年龄、用户性别、地理位置这四个维度，我们还可以对浏览器、用户职业、用户活跃度、用户行为偏好等各种各样的维度进行分析，甚至可以综合考虑以上维度，在某个特定范围内挖掘热点事件，或者根据语言习惯去寻找出某个特定的人群。或许这听上去太过理想化，不过我坚信，有了合适的算法，这些想法终究会被一一实现。

顾森

网名Matrix67，北京大学中文系应用语言学专业学生，数学爱好者。2005年开办数学博客http://www.matrix67.com，至今已积累上千篇文章，已有上万人订阅。

责任编辑：卢鹄翔（ludx@csdn.net）

Facebook Folly源代码分析（下）

文 / 李凯

本文将继续分析Folly库的两个多线程并发数据结构：队列（ProducerConsumerQueue）和跳转表（ConcurrentSkipList），其中队列使用原子操作保证单读者单写者情况下，无锁的多线程安全；而跳转表则使用小粒度的加锁方式，实现多线程读写情况下的高并发操作。

ProducerConsumerQueue.h

ProducerConsumerQueue是一个简单的无锁队列，特点是需要预先确定大小，推入和弹出各只允许一个线程。类在构造时申请固定长度的内存用于存储对象数组，写入时在空闲位置的内存空间上构造对象。

由于使用了C++0x的变长模板参数特性，它的写入接口可以支持任意数量和类型的参数，既可以传入对象进行拷贝构造，也可以传入其他构造函数参数进行构造。使用writeIndex和readIndex分别作为读写位置的数组下标，writeIndex为下次写入时的可写位置，readIndex为下次读取时的可读位置，writeIndex和readIndex随着每次写入或读取加1，加到数组大小后则回滚到0。其读写操作流程描述如下。

■ 写入操作

1. 获取writeIndex加1后的值nextWriteIndex，如果大于数组大小则改为0；

2. 判断nextWriteIndex是否等于readIndex，如果相等，表示队列已满，否则队列未满；

3. 如果队列未满，则将数据写入writeIndex标记的位置，然后修改writeIndex为nextWriteIndex的值。

■ 读取操作

1. 判断readIndex的值是否等于writeIndex，如果相等，表示队列已空，否则队列非空；

2. 如果队列非空，则读取readIndex标记位置上的数据；

3. 将readIndex的值加1后进行判断，如果大于数组大小则改为0，将新值赋给readIndex。

ProducerConsumerQueue的实现方式比较巧妙，但我认为还有两处有待改进。

首先，内存管理比较简单。在构造时一次性分配连续地址空间，即使使用者能够确保在多数情况下不会将队列用满。但因为这种循环队列的方式不会优先使用已被读出的空间，随着不断写入和

读出，最终所有的空间都将被用尽，即构造时申请了多大的虚拟内存空间，最终也将占用一样大小的物理内存。

其次，虽然对单写者和单读者的限制，使得其实现比较简单，但这种限制会造成应用场景变得很局限。无论是内存管理、网络框架、线程池等应用，对Queue最起码的要求是能支持单读多写或多读单写。

在现有ProducerConsumerQueue实现的基础上进行不算复杂的改进，即可突破上述限制。

1. 指针与内存的分开管理，使用可复用的内存池管理对象空间，使用Queue存储对象指针。写入对象时，从内存池分配内存构造对象，将对象指针加入队列；读取对象时，从队列中弹出指针，读取对象后，释放回内存池。由于一个指针占用内存很小，队列本身所占用的内存也将随之变小。

2. 如图1所示，基于writeIndex和readIndex两个数组下标的方式，多读多写队列的实现仍然使用定长数据来存储对象。使用writeIndex表示下次写入时的可写位置，使用readIndex表示下次读取时的可读位置。

而与ProducerConsumerQueue不同的是，writeIndex和readIndex一直保持增加，而不是在增加到数组大小时回滚为0。因此，通过它们访问数组时，需要先对数组大小取模。另外一个不同之处是，定长数组的元素，除了保存对象本身，还要额外保存一个名为curIndex的值，它表示当时写入这个元素时所用的writeIndex值，是在多线程情况下数据有效的标志。

3. 判断队列当前已使用大小的方法比较简单，由于writeIndex和readIndex只增加不回滚，因此它们的差值即为队列当前大小。

改进的读写操作流程描述如下。

■ 写入操作

1. 保存writeIndex的当前值为curWriteIndex；

2. 原子的CAS操作循环内容：判断队列未满，将writeIndex加一；

3. 将数据写入curWriteIndex所标记的位置，然后将这个位置上的curIndex值修改为curWriteIndex。

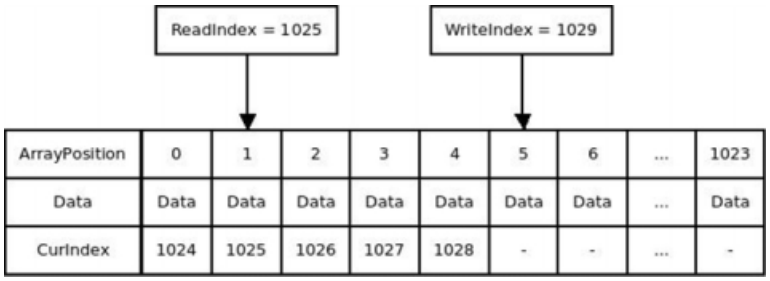


图1　判断队列当前已使用大小的方法

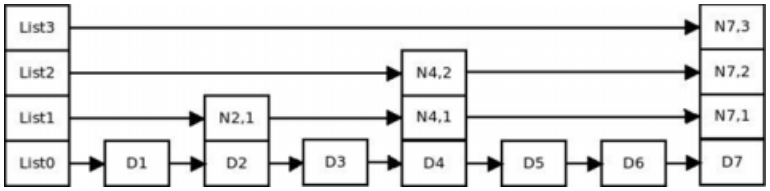


图2　跳转表的设计与实现

■ 读取操作

1. 保存readIndex的当前值为curReadIndex；

2. 原子的CAS操作循环内容：判断队列非空；curReadIndex所标记位置上的curIndex与curReadIndex相等，将ReadIndex加一；

3. 读出curReadIndex所标记位置的数据。

ConcurrentSkipList.h

ConcurrentSkipList是一个可读写并发操作且互不影响的跳转表，它实现了lock-free并且wait-free的并发读，以及很小粒度加锁的并发写（由于使用了tryLock并重试的方式获取写锁，因此也可以认为写操作是lock-free的）。内部使用大粒度的引用计数管理内存池，用于节点内存的申请、释放和重用。对外提供了多线程安全的insert、erase、find和iterator接口。

跳转表的设计和实现参考了注3中的论文，设计结构如图2所示。

使用链 表将所有数据按照排序顺序连接起来，每个data节点即为一条数据记录，每个节点除保存数据和指向下一个节点的指针外，还会包含随机大小的next指针数组，在这个数组中的第N个next指针表示跳转表的第N级索引链的next指针，如图2共有3级索引链，第0级索引链即为所有

数据的链表，第1级索引链有4个节点，第2级索引链有3个节点，第3级索引链有2个节点。插入、删除或查询数据时，从最高级的索引节点开始一级一级向下查询，比如要查询D5节点，查询路径为List3→List2→N4,2→N4,1→D4→D5。

在插入新节点时，使用随机的方式决定是否将新节点插入更高几级的索引链表，但随着层级的增加，产生索引的概率也随之减小（算法参考了注5中在Java的实现）。如图2所示，在插入D4这个节点时得到的随机索引等级为2，因此，这个结点将插入第1~2级的索引链表中；而D7节点得到的随机索引等级为3，因此它要插入到第1~3级的索引链表中。

在删除节点时，需要将其从索引链表中同时删除，因此先将其状态标记为正在删除，然后操作链表，最后将节点内存释放回内存池。由于在并发情况下查询过程并不会加锁，可能出现其他线程仍然访问到被删除节点的情况，因此需要保证节点释放回内存池后，必须在后续没有其他线程访问的情况下，才能复用或释放回系统。

ConcurrentSkipList使用一个简单的机制来处理这个问题，每次访问跳转表（包括读写删除）前，对内存池引用计数加1，访问完成后将引用计数加1，当引用计数减到0时，将之前释放的内存复用或释放回系统。这种机制的问题是在大压力读写删除的情况下，可能无法及时将已释放的内存复用，而对于删除操作比较多的应用可以考虑使用诸如HazardPointer（参见注4）的复杂技术管理内存申请和释放。每个数据节点包含了以下几个数据域。

1. data_是用户数据对象。
2. spinlock_是互斥锁标记，使用tryLock方式加锁，而非阻塞等待。
3. flags_标记了三种节点状态。

IS_HEAD：链表头结点，全局只有一个。

FULLY_LINKED：节点已被加入所有需要加入的索引链表，是可用的标志。

REMOVING：节点正在被删除，已不可用，遇到此标记需要重试流程或返回失败。

4. skip_[0]是next指针数组，至少包含一个元素（即所有数据的链表），从第1个元素之上的第N个元素是第N级索引链表的next指针。

具体的数据查询、插入和删除操作流程和一些细节分析如下。

■ 查询操作

1. 从Head节点的最高层索引的next指针开始向低层索引的next指针遍历，直到遇到小于Key的最大节点（即第Ln层索引next指针指向的节点），将这个next节点记为N；如果遇到等于查询Key的情况，则直接返回。

2. 以N节点和Ln层索引作为遍历起始，按照第1步的方法继续向低层索引遍历查询。

在查询操作的代码中，实现了名为findNodeDownRight和findNodeRightDown两个查询函数，分别表示优先向下遍历同一节点的多级索引，还是优先向右遍历同一级索引上的链表。findNodeDownRight在测试中性能要稍好于findNodeRightDown。

但根据skiplist先查询高级索引，再查询低级索引的原则，先要查询同一级索引后才能转向下一级索引，必然是先向右后向下的顺序。而findNodeDownRight函数中，下面这个for循环也说明，只要高一级索引需要向右遍历，就不会转向低一级的索引。

```
for (; ht > 0 && less(data, pred->skip(ht - 1));
     --ht) {}
```

因此，可以认为两个函数的查询方向都是先向右后向下的，只是实现方法不同，至于性能上的差别，Folly中的注释认为前者更利于CPU局部缓存优化。但我认为还有可能在于赋值操作node = pred->skip(*)，在不需要向右遍历同级索引的情况下变得没有意义。

■ 插入操作

1. 运行查询操作，找到数据要插入的位置。在查询过程中，记录每一级索引查询路径上的最后一个节点（称为predecessor）和这个节点的next节点（称为successor），比如要插入的数据在图示位置中的D4和D5之间，那么需要记录的predecessor节点和successor节点依次为(List3

N7,3)、(N4,2 N7,2)、(N4,1 N7,1)、(D4, D5)。

2. 如果在查询过程中找到了要插入的数据，那么存在两种失败的情况：如果数据所在节点被标记为正在删除，那么重新从第1步开始插入流程；如果数据所在节点被标记为非正在删除，那么自旋等待节点的FULLY_LINKED标记变为true后返回插入失败，失败原因是结点已存在。

3. 根据随机算法，得到新插入节点要插入索引链表的等级值，多数情况下随机结果为0，即不需要加入任何一级的索引链表。如果随机结果为1，表示需要加入第1级索引；如果随机结果为2，表示需要加入第1和第2级索引，以此类推。如图2中的D4节点，它在插入skiplist时，随机结果为2，因此它同时存在于第1级和第2级索引链表。随机结果的范围在[0, 当前索引等级个数]之间产生。

4. 对第一步中得到的多个predecessor节点，从第0级开始的N个节点执行tryLock，如果有失败则将已加的锁解开，然后重新从第1步开始插入流程，同一个节点在不同级别上不重复加锁。N为第3步中得到的随机值，因为不需要加入大于N的索引链表，就不需要修改大于N级的predecessor节点，也就不需要对它加锁。

例如要插入的数据在图示位置中的D4和D5之间，如果随机值为0、1或2，则只需要对D4节点加锁；如果随机值为3，则需要对Head、D4节点加锁。并发情况下，所有线程加锁的方向都是从大到小，因此不存在死锁风险。

5. 对一个Predecessor节点加锁后需要检查。首先，节点的删除状态如果被标记为正在删除，说明有其他线程正在删除Predecessor节点，那么需要重新从第1步开始插入流程；其次，节点的next是否仍是刚才保存的successor节点，如果不是，说明过程中有其他线程插入或删除节点，那么需要重新从第1步开始插入流程。

6. 构造要插入的节点，将其加入第0级链表和更高级的索引链表后打上FULLY_LINKED标记。

7. 对上述多个predecessor节点解锁。

8. 判断如果当前节点总数过多，则增加一级索引，不修改当前数据，而是在下次插入数据时，生成随机数的范围加1。

■ 删除操作（基本与插入过程类似，但有几处值得注意）

1. 加锁predecessor节点的个数由要删除数据节点的索引级别决定，如图2中的D2节点，如果需要删除，则需要对Head和D1节点加锁；而如果要删除D3节点，则只需要对D2节点加锁。

2. 判断节点是否可删除时要判断节点的索引级别是否与查询过程中遍历得到的索引级别一致，如果不一致则可能是节点正在被加入或摘除。

注释

注1: Adaptive Locks: Combining Transactions and Locks for Efficient Concurrency

注2: Using Spin-Loops on Intel® Pentium® 4 Processor and Intel® XeonTM Processor

注3: A Provably Correct Scalable Concurrent Skip

注4: Hazard Pointers: Safe Memory Reclamation for Lock-Free Objects

注5: Doug Lea. ConcurrentSkipListMap. In java.util.concurrent

李凯

淘宝核心系统部存储组技术专家，花名郁白。2007-2010年曾参与百度分布式文件系统研发，2010年至今参与淘宝0ceanbase项目的研发。

责任编辑：卢鹃翔（ludx@csdn.net）

谁动了我的句柄

文 / 张银奎

下班到家，有一份快递摆在案头，一看地址，来自千里之外。打开层层包装，是一张光盘，装光盘的纸袋上七扭八歪地写了两行字：“程序崩溃，偶尔出现；困扰已久，务请帮忙”。明白了，是朋友要我帮忙分析的转储文件到了。

无效句柄

都云时代了，为什么还要用光盘快递呢？原因是文件很大，压缩后还1GB有余。我唤出WinDBG，打开转储文件，片刻之后，WinDBG给出了目标系统的概况、产生转储文件的时间、系统的启动时间和进程的启动时间。在这些概要信息之后，WinDBG给出了如下几行信息：

```
(bc8.ab0): In.valid handle - code c0000008 (first/second chance not available)
ntdll!NtGetContextThread+0xa:
00000000`773b0a2a c3     ret
```

无效句柄错误代码为c0000008，使用!error命令观察这个错误码，得到的信息很一致。

```
!error c0000008
Error code: (NTSTATUS) 0xc0000008 (3221225480) - An invalid HANDLE was specified.
```

c0000008之后括号里的话，是说不知道此刻是第一轮分发异常还是第二轮，通常转储文件中都不包含这个信息。

总的来说，第一句话提示我们曾经有一个无效句柄异常发生，后面两句可能是产生转储文件时调用了GetThreadContext API。不过这几行信息的真实性还有待进一步分析，可能是子虚乌有，也可能是一语道破天机。

回到异常现场

执行k命令观察栈回溯，乱七八糟，只有开头的一个栈帧有函数名，也就是上面提到的NtGetContextThread，其他全是长短不等的16进制数。

当在转储分析时遇到这种情况，首先想到的化解方法就是想办法切换到发生异常时的上下文，因为当前的执行上下文可能是写转储文件时的上下文，花时间得到栈回溯也不是希望看到的异常现场。

如何切换到异常现场呢？因为异常对于调试有着特殊的重要意义，所以转储文件专门定义了一种用来记录异常信息的数据块，称为异常数据流，是一个MINIDUMP_EXCEPTION_STREAM结构（详见《软件调试》12.9节“用户态转储文件”）。当使用WinDBG分析转储文

件时，WinDBG有一条专门的元命令来访问这个数据块，叫.ecxr，执行这条命令后，WinDBG便会读取异常数据块，并根据其中的信息做“时光挪移”，切回到异常现场。

因为系统软硬件都对异常有特殊的支持，一个异常从发生那一刻起，便有专门的数据结构为其建立档案，全面记录异常发生时的详细信息，这个异常结构会始终伴随异常的分发和处理过程，当产生转储文件时，负责转储的系统函数会将这个结构原原本本地写到转储文件中。因此，在分析转储文件时，异常上下文中的信息具有非常高的可信度，值得我们认真体会。

信息的上半部分是发生异常时的寄存器内容，其中RAX和RBX的内容恰好是我们前面看到的异常代码——c0000008。接下来是标志寄存器和段寄存器。最后两行是RIP寄存器（程序指针）所指向的位置和汇编指令。因为已设置了符号路径，所以WinDBG已经帮我们查找出了对应的函数，即NTDLL.dll中的RtlRaiseStatus函数。看来是系统函数触发的异常。

何处炸“雷”

我经常把抛出异常比喻为投出一颗手雷，一旦投出，必须紧急处理，不然整个进程可能就结束“生命”了。回到我们的问题，是哪里的代码调用RtlRaiseStatus来投“雷”呢？很自然地想到执行kn命令来观察栈回溯，下面是部分结果：

```
0:012> kn
 *** Stack trace for last set context
 # Child-SP   RetAddr   Call Site
00 0`5df7f530 0`773c78e9
ntdll!RtlRaiseStatus+0x18
01 0`5df7fad0 1`40007d7a ntdll! ??
::FNODOBFM::`string'+0xa3ee
02 0`5df7fb00 1`4000c09e XxModule+0x7d7a
```

三个星号开始的那句话是在提示我们现在的观察点是因为前面执行.ecxr命令而“时光倒流”过的时间点，并不是当前线程的“最新”时间点。接下来是一个个的栈帧，我只摘取了前三个栈帧，以节约篇幅；删去了64位数中无效的0，以减少换行；而且将第三个栈帧中包含的真实模块名替换成XxModule，目的是将真名隐去，以免有人对号入座。

仔细看这三个栈帧，0号是抛出异常的RtlRaiseStatus，1号的名字乱七八糟，稍后细说，2号是XxModule中的用户代码，看起来是用户代码调用了某个API后，这个API又调用了RtlRaiseStatus。因为RtlRaiseStatus是没有公开的函数，用户代码不大可能直接调用它。

那么调用的是哪个API呢？答案应该在1号栈帧中，但kn命令给出的1号栈帧函数名信息极其模糊，没有正面回答，只是说这个函数相距ntdll! ??::FNODOBFM::`string'这个字符串的距离有0xa3ee个字节这么远。

现在如何是好？软件调试如同警察破案，讲究的多管齐下，一条线索断了，便换个角度找另一条。kn命令做栈回溯是沿着从栈顶到栈底的方向依次找函数的父函数，那么我们不妨反其道而行之，沿着从栈底向栈顶的方向，从父函数中找子函数的名称。也就是在2号栈帧的函数中找它调用了哪个子函数。

轻敲键盘，发出ub 01`40007d7a命令，成功得到01`40007d7a前的8条指令，最邻近01`40007d7a的一条果然是call指令：

```
call    qword ptr [XxModule+0x520b8
(00000001`400520b8)]
```

还是没有直接给出函数名，因为这里调用的是一个函数指针，不要紧，可以使用如下命令来读出函数指针指向的内容，并查找它对应的名称：

```
0:012> ln poi(0000001`400520b8)
(00000000`773ad070) ntdll!RtlLeaveCriticalSection
| …
Exact matches:
    ntdll!RtlLeaveCriticalSection = <no type
information>
```

终于得到函数名了，原来是RtlLeaveCriticalSection，没有这个API啊，其实，它就是LeaveCriticalSection API的“真身”。当我们在应

用程序调用LeaveCriticalSection API时，运行时便会执行RtlLeaveCriticalSection。

代码“流放”

看来是XxModule中的用户代码调用LeaveCriticalSection时触发了异常。为了能让热门的代码相邻，优化工具需要将执行概率很低的代码块移开，以便腾出地方，我们不妨将这些被优化工具移动到“边远”地带的代码块称为被“流放”的代码。用作错误处理的代码执行概率较低，是首选的“流放对象”。比如在本例中，调用RtlRaiseStatus函数的错误处理代码块在地址773c78e9附近，它自己的函数入口地址为773ad070，二者之间的距离为：

```
0:012> ?? 0x773c78e9-0x773ad070
int 0n108665
```

RtlLeaveCriticalSection函数的长度根本没有这么长。相距这么远，完全是错误代码被“流放”的结果。

流放低概率代码有利于提高物理内存的利用率和软件的执行速度，但对调试来说，却不是一件好事。因为“代码流放”动作通常是在编译后使用优化工具完成的，优化工具只调整可执行文件，不会调整调试符号（至少目前还是这样），这便会导致调试器无法找到被流放的代码所属的本来函数，只能为其就近寻找参照物。在上面的栈回溯中，?? ::FNODOBFM::`string'就是调试器找到的参照物，这个参照物的质量实在不怎么样，难以读出，没什么意义。这个信息的价值非常有限，有点像是说，“在距离马马虎虎岛8000海里处”，而且没人知道马马虎虎岛在哪儿。

事出何因

经过一番“穷追猛找”，我们终于搞清楚了，发生“爆炸”的地方是RtlLeaveCriticalSection函数的“被流放”代码。接下来的问题是为何发生爆炸呢？搞清这个问题也不是件容易的事，因为我们并没有RtlLeaveCriticalSection函数的源代码，因此要找出其中原委，须得再费一番心思。如何动手呢？不妨就从调用RtlRaiseStatus函数的地方反向反汇编。先从栈回溯中提取RtlRaiseStatus函数的返回地址773c78e9，将其作为参数执行ub命令，即：

```
0:012> ub 773c78e9
ntdll!LdrGetProcedureAddress+0xf68e:
773c78ce 4533c9    xor     r9d,r9d
…
773c78dd e93657feff jmp     ntdll!CsrAllocateMessagePointer+0x2b8 (773ad018)
773c78e2 8bc8      mov     ecx,eax
773c78e4 e8878b0600 call    ntdll!RtlRaiseStatus (77430470)
```

这个结果怎么样呢？最下面的call指令证明这里的确调用了RtlRaiseStatus函数，其上的mov ecx, eax是将EXA寄存器中的内容赋值给ECX，作为参数传给RtlRaiseStatus函数。在异常现场，我们看到EAX寄存器的值仍为c0000008，即无效句柄的异常代码，看来导致崩溃的无效句柄异常就是从这里发起的，EAX寄存器中还残留着痕迹。

接下来的问题是，CPU是怎么执行到这里来的呢？MOV语句上面是一条绝对跳转，这意味着CPU绝对不是从我们看到的这串指令的上面顺序执行下来的，而是从其他地方飞到MOV语句这里的。这个原因让我们省略掉了jmp语句的内容，因为与我们的问题可能完全不相干。那么CPU是从哪里飞过来的呢？尽管CPU内部已有分支监视机制可以帮助记录CPU的完整执行轨迹，但是通常是不开启的，而且我们现在是分析转储文件，这个信息更无从得到。解决的办法只能是继续做反汇编，从汇编代码中寻找跳转到这条MOV指令（地址为773c78e2）的跳转语句。执行uf RtlLeaveCriticalSection让WinDBG帮我们反汇编整个RtlLeaveCriticalSection函数，然后在其中搜索MOV指令的地址773c78e2。不消眨眼工夫，反汇编的结果便出来了。浏览一下很容易看出，因为做过优化，所以代码块的顺序很混乱，不去管它，只管搜索773c78e2，果然找到，且只有一处（作为跳转目标），是条绝对跳转（jmp）语句：

```
773c0caf e92e6c0000    jmp     ntdll!LdrGetProcedureAddress+0xf6a2 (00000000`773c78e2)
```

看看这条语句的上一条指令，居然是一条用作函数返回的ret语句，看来又是从其他地方跳到这条jmp语句的，继续搜索jmp语句的地址773c0caf，也只有一处。这块代码值得细看，摘录如下（省去机器码）：

773ad011 xor edx,edx

773ad013 call ntdll!ZwSetEvent (773afe30)

773ad018 test eax,eax

773ad01a jns ntdll!RtlLeaveCriticalSection+0x34 (773ad0a4)

773ad020 jmp ntdll!LdrGetProcedureAddress+0x8a6f (773c0caf)

绝对跳转语句（jmp）之上是一条跳转跳转语句，跳转的依据是上面的test语句，test语句的操作数是EAX寄存器，EAX的内容哪里来呢？就是上面ZwSetEvent函数的返回值。综合这几条指令，就是调用ZwSetEvent内核服务，然后判断它的返回值，如果返回值不小于0就转移到773ad0a4，如果小于0，便执行下面的绝对跳转语句，转去错误处理了。这正符合包括ZwSetEvent在内的大多数Windows系统服务的返回值约定习惯：0代表成功（S_OK），失败则返回相当于负数的错误码。这与Win32 API中公开的SetEvent函数的返回值约定有所不同，API层为了简化，只返回TRUE或者FALSE，但可以通过GetLastError返回失败的错误码。事实上，后者只是对前者的简单封装。

```
BOOL WINAPI SetEvent(
  __in  HANDLE hEvent
);
```

至此，我们又进展了一步，了解到“爆炸”的原因是调用ZwSetEvent内核服务，这个内核服务返回了错误，异常现场看到的无效句柄异常代码应该就是这个内核服务所返回的，因为调用这个服务后，经过两个绝对跳转就调用RtlRaiseStatus走上绝路了，而提交给RtlRaiseStatus的“罪名”就是ZwSetEvent的返回值，放在EAX寄存器中，一直没有动过。

哪个句柄

大家都知道，句柄就是个整数，是放在内核中的内核对象的一个用户态代号。一个程序通常会使用很多内核对象，因此有很多个句柄。接下来的目标是找出是哪个句柄出了问题。也就是找出调用ZwSetEvent内核服务时的参数。调用动作已过去很久，如何找呢？

一种思路是像无意中看到RtlRaiseStatus函数的参数取值那样，从异常现场的寄存器中寻找。继续看一下调用ZwSetEvent函数时的汇编代码，可以发现，它的参数是使用RCX寄存器来传递的。异常现场的RCX取值为000000005df7f5f0，根据多年编程和调试的经验，一眼看去，这就不像是一个句柄，句柄值一般没有这么大。看来时过境迁，RCX寄存器中的内容已经不是调用ZwSetEvent时的句柄了。

这条线索走不通了么？没有。寄存器中的内容很容易变化，与之相对而言，比较稳定的是内存中的内容。也就是说，我们可以追踪感兴趣的寄存器内容是否来自内存或者向内存中写过，然后从相关的内存中寻找当时的记忆。

按着这条思路首先追索RtlLeaveCriticalSection函数，句柄来自RBX寄存器指针的偏移0x18处，RBX的值来自RCX，也就是RtlLeaveCriticalSection的第一个参数。结合LeaveCriticalSection API的函数原型：

```
void WINAPI LeaveCriticalSection(
  __inout  LPCRITICAL_SECTION lpCriticalSection
);
```

可以知道这个参数的类型是一个指向CRITICAL_SECTION结构的指针，可以在SDK中找这个结构的定义：

```
typedef PRTL_CRITICAL_SECTION LPCRITICAL_SECTION;
```

上溯到RtlLeaveCriticalSection的父函数寻找参数的来源。使用前面提到的方法，反向反汇编RtlLeaveCriticalSection的返回地址：

lea rcx,[rdi+18h]

call qword ptr [XxModule+0x520b8 (00000001`400520b8)]

继续从父函数中追索，rdi来自rcx，而且让我们倍感欣慰的是，rcx的值曾经赋值到内存中：

mov qword ptr [rsp+38h],rcx

根据经验，这是将一个对象指针赋值到栈上的指

针变量，太好了！栈上的内容有较好的持久性，尤其对于尚未返回函数所使用的栈帧来说更是如此。

因为栈上的内容是浮动的，所以索引栈上的变量时一般都使用寄存器作为参照物，在32位程序中，经常使用EBP，有时也使用ESP，在64位程序中，通常总是使用RSP作为参照物。接下来就是如何找到这个函数执行时使用的RSP值，这个信息在前面执行过的kn命令中就可以找到，Child-SP列的内容便是，找#2栈帧对应的值，即5df7fb00。

检验一下，首先观察栈帧中偏移0x38处的指针变量值：

```
0:012> dq 5df7fb00+38 l1
00000000`5df7fb38  00000001`4006fbb0
```

再观察指针所指向对象偏移18处的关键区结构：

```
0:012> dt _RTL_CRITICAL_SECTION
0000000014006fbb0+18
XxxxModule!_RTL_CRITICAL_SECTION
   +0x000 DebugInfo        : 0x00000000`0027cc80 _
RTL_CRITICAL_SECTION_DEBUG
   +0x008 LockCount        : -3
   +0x00c RecursionCount   : 0
   +0x010 OwningThread     : (null)
   +0x018 LockSemaphore    :
0x00000000`00000460
   +0x020 SpinCount        : 0
```

看起来非常像关键区，如果想进一步验证，可以观察结构体中的DebugInfo指针所指向的调试信息结构。

```
0:012> dt 0027cc80 _RTL_CRITICAL_SECTION_
DEBUG
DebugInfoDLL!_RTL_CRITICAL_SECTION_
DEBUG
   +0x000 Type             : 0
   +0x002 CreatorBackTraceIndex : 0
   +0x008 CriticalSection  : 0x00000001`4006fbc8 _
RTL_CRITICAL_SECTION
    +0x010 ProcessLocksList : _LIST_ENTRY [
0x00000000`0027ccd0 - 0x27cc50 ]
   +0x020 EntryCount       : 0
   +0x024 ContentionCount  : 1
   +0x028 Flags            : 0
   +0x02c CreatorBackTraceIndexHigh : 0
   +0x02e SpareWORD        : 0
```

调试信息结构中的CriticalSection字段指向对应的关键区结构，它的值正好与关键区结构的地址1`4006fbc8一致。

观察进程里的其他线程，有一个线程正在等待进入关键区，观察它的寄存器上下文，rbx=000000014006fbc8，正是同一个关键区。

至此，可以很肯定地说，我们找到了导致问题的句柄值（0x460），而且找到了使用这个句柄的关键区对象。

结局

当我们分析出以上结论时，就把这个结果先告诉了急于想知道进展的“求救者”。出乎我的预料，他们根据我提到的句柄值，搜索日志文件，居然就找到了酿成错误的源代码，把问题解决了。我听到消息，也很是高兴。因为要继续寻找答案，光靠转储文件就有些不够了。接下来的最佳方案是使用应用程序验证器，启用Windows系统的句柄跟踪功能，让验证模块自动跟踪和校验进程中的所有句柄使用情况，并在发现问题时立刻触发“验证停止”并发出提示。

张银奎

《软件调试》作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。

责任编辑：卢鹃翔（ludx@csdn.net）

有奖问答：

上期问题：WRL是什么含义？WRL的全称为Windows Runtime Library，是用来封装WinRT底层接口的C++类库，旨在提高开发效率，与ATL的思想类似，推荐阅读channel9中的采访：GoingNative 2: C++ at BUILD, Windows Runtime Library(WRL), Meet Tarek and Sridhar。

本期问题：关键区结构中的SpinCount字段用来实现什么功能？

编者说明：

- 编辑信箱：ludx@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
  1) 姓名
  2) 工作单位或学校
  3) 电话联系方式
  4) 邮寄地址
  5) E-mail地址
- 解答提交时间，最好早于当月15日。

Mac OS X背后的故事（十一）

Mac OS X文件系统的来龙去脉（下）

文 / 王越

由于各种缺点，干掉HFS+势在必行，然而用什么取代HFS+呢？苹果开始秘密研发下一代的文件系统——ZFS，然而在诸多因素的干扰下，Mac OS X的ZFS支持却只是昙花一现，未来文件系统之路将走向何方？

文件系统的新时代——ZFS

为了代替HFS+，苹果开始为研发下一代文件系统招兵买马，准备大干一场。但这时Sun公司的工作让苹果的员工们为之一振。

2004年，Sun公司发表了其杰出的文件系统ZFS。这是一个128位的文件系统，本为Solaris操作系统开发，于2005年10月31日并入了Solaris开发的主干原始码。后成为一个使用CDDL协议条款授权的开源项目。

ZFS是一个具有高存储容量、文件系统与卷管理概念整合、崭新的磁碟逻辑结构的轻量级文件系统，同时也是一个便捷的存储池管理系统。

ZFS的一个重大特点就是拥有大容量。ZFS是一个128位的文件系统，这意味着它能存储1800亿亿（18.4×10^{18}）倍于当前64位文件系统的数据。ZFS的设计如此超前以至于这个极限就当前现实而言可能永远无法遇到。项目领导Bonwick曾说：“要填满一个128位的文件系统，将耗尽地球上所有存储设备，除非你拥有煮沸整个海洋的能量。”假设每秒钟创建1000个新文件，达到ZFS文件数的极限需要约9000年。

此外，ZFS的一个重要指导思想是不单单去做一个文件系统，而是实现一套完整的卷管理方案。不同于传统文件系统需要驻留于单独设备或者需要一个卷管理系统去使用一个以上的设备，ZFS建立在虚拟的被称为“zpools”的存储池之上。每个存储池由若干虚拟设备组成。这些虚拟设备可以是原始磁碟，也可能是一RAID1镜像设备，或是非标准RAID等级的多磁碟组。于是zpool上的文件系统可以使用这些虚拟设备的总存储容量。

有了卷管理方案后，ZFS走得更远，加入了快照和克隆等实用的文件系统功能。当ZFS写新数据时，包含旧数据的块被保留，磁盘只写入修改过的那部分数据块。所以快照的建立非常快，只存储两个快照间的数据差异，因此快照也是空间优化的。克隆指两个独立的文件系统共享一些列的块。当任何一个克隆版本的文件系统被改变时，只创建改动的数据块，因此非常快速，也占用少得多的空间。

而ZFS最大的贡献在于它是第一个支持写入时复制功能（COW，copy on write）的文件系统。所有文件系统中的块都包括256位的校验值。含有活动数据的块从来不被覆盖；而是分配一个新块，并把修改过的数据写在新块上。所有与该块相关的元数据块都被重新读、分配和重写。因此，当一个数据写入时发生了任何意外错误，原先的数据依然可以被访问，且文件系统知道哪个操作出了错误而没有完成。ZFS的快照和克隆正是因此项技

术而得以实现。

ZFS对于用户而言，界面友好。先前Unix的卷管理非常烦琐，FreeBSD因此还建了一套宏伟的框架，给逻辑卷管理做深层次的抽象。而ZFS文件系统自带卷管理方案，几乎所有烦琐复杂的操作都能在一两条命令内完成，我用传统的卷管理工具已有近十个年头，第一次使用ZFS时，完全被其易用性震撼，所以我毫不犹豫地把手头所有的服务器迁移到了ZFS。

苹果看到ZFS的移植工作在短短一年中取得突破性的进展，甚为欢欣，打算使用ZFS作为下一个操作系统的文件系统。

由于ZFS各种美好，加上其开源性质，所有的操作系统都想支持它。Solaris、OpenSolaris项目一直作为标准实现供其他系统参考。Pawe Jakub Dawidek把ZFS移到FreeBSD，并在2009年进入了FreeBSD 7，作为FreeBSD第七版最耀眼的三项功能之一（另一项功能是我们先前提到的ULE，以及Sun DTrace的移植工作）。NetBSD在2009年正式收纳ZFS。Linux则麻烦得多，因为Linux内核的协议GPL是个和很多协议都水火不容的奇葩协议，ZFS分发所采用的CDDL和GPL会产生冲突，所以一方面FUSE提供了用户空间层面的支持；另一方面，由Oracle牵头，专为Linux开发Btrfs，事实上就是一个ZFS的山寨版，可惜折腾了几年，Oracle自己又把Sun收购了，且到我撰写此文时Btrfs依然没有正式的稳定版本发布。

昙花一现的ZFS梦

刚才提到，苹果在招兵买马，雇员工开发新一代的文件系统，而Chris Emura（Apple CoreOS 的文件系统开发经理）及Don Brady（先前提到，此人领导HFS+的开发）两个富有经验的文件系统开发者却被衣服一样晾在了一边无所事事。2006年，刚刚提到的Pawe Jakub Dawidek正在往FreeBSD迁移Sun的ZFS，这项工作立刻引起了Chris Emura及Don Brady的高度兴趣。由于ZFS在Unix系统高度的可移植性，加上Mac OS X本就是FreeBSD的近亲，闲得发慌的两人立即打算往Mac OS X移植ZFS。在2007年4月6日，FreeBSD的移植宣告完成，等待合并进主干。一周后，两位苹果员工亦成功地完成了Mac OS X的移植。

苹果一看两人的ZFS的移植工作大有前途，立即跟进。2007年的苹果全球开发者大会上，苹果让Chris Emura及Don Brady举办了一场小型讲话，介绍Mac OS X对ZFS的支持。这场讲话先前并没有在官方声明中告示，但讲话的报告厅依然挤满了听众。随后ZFS移植的源码在Mac OS Forge公布。在最终版的Mac OS X 10.5带有试验性的ZFS只读支持，以命令行方式提供。用户可以挂载ZFS的存储池，并对池中的文件系统进行读取操作。

苹果一直使移植并使用Sun的关键技术，除了Java以外，Mac OS X 10.5的Xcode套件也加入了DTrace的支持，并提供了一个好用的图形界面Instruments让开发者更方便地调用DTrace。ZFS除了解决HFS+的所有问题，提供安全可靠的文件系统基础外，还可以简化苹果许多软件的实现。例如前文提到的Mac OS X 10.5的Time Machine，实现颇为烦琐，依赖于给HFS+提供新功能，功能层也需要增加很多的和备份相关的代码。而ZFS默认就支持快照，将大大简化Time Machine的实现，并使该功能更稳定可靠。事实上在2008年11月25日，Sun发布了OpenSolaris 2008.11版，其中给GNOME的Nautilus增加了一个使用ZFS的快照功能的图形界面插件名为Time Slider，和苹果的Time Machine提供了非常相近的功能，我在使用后感觉不错。

因此在WWDC 2008上，Snow Leopard被提出，其中一项很重要的卖点就是对ZFS的完整的读写支持。在Mac OS X的服务器版，苹果也将提供一套图形界面工具来方便维护人员管理ZFS存储池。在当时的Snow Leopard Server主页上，苹果声明ZFS将作为一项主推功能。

但好景不长，一年后的苹果开发者大会时，ZFS相关的内容被悄悄从任何公开的文档、网站、发布会中撤下，没有给出任何的理由。Mac OS Forge上的ZFS代码和页面也被苹果移除。外界有很多对此的猜测，但没有任何猜测得到苹果官方的或

是哪怕离职员工的证实。

猜测之一是当时Sun刚被Oracle收购，而Oracle长期投资ZFS的竞争产品Btrfs。因此苹果觉得ZFS的前途不甚明朗。

猜测之二是ZFS的关键技术Copy On Write有专利问题，NetApp声称他们拥有COW的专利因此在起诉Sun，苹果不想在当中冒风险。

猜测之三是ZFS和苹果的XNU内核有协议冲突。我虽然不学法律，但我认为这个说法不完全对，因为ZFS和DTrace一样，是以CDDL发布的开源软件，既然DTrace可以无后顾之忧地加入到XNU中，ZFS也没有理由不可以。事实上，除了Linux这种少数使用GPL这类奇葩协议的内核，大多数系统的协议都不和CDDL冲突。FreeBSD也好，Mac OS X 10.5也罢，都把ZFS加入内核发布。

但事实上，如果把三种猜测并在一起，我们可以看到一个更全局的可能性：对于猜测之二，苹果可能并非想使用CDDL，而是想从Sun买下一个私有的协议，这样一来，Sun不但提供更好的技术支持，出了问题（比如猜测二中的专利问题）也可以让Sun为自己背黑锅。结果Sun可能和苹果价格谈不拢，加上猜测之一提到的Sun大势已去，让苹果觉得还不如自己造个轮子来得方便。Sun公司开发ZFS的主力Jeff Bonwick虽不能提供详细的信息，但他基本证实了这种说法。

无论如何，Mac OS X的ZFS支持，如昙花一现般消失了。

未来文件系统之路走向何方

虽然Mac OS X的ZFS支持被砍了，开源社区依然想继续开发Mac OS Forge先前版本的移植。如MacZFS项目不遗余力地给Mac OS X 10.5~10.7提供ZFS读写支持。Don Brady在苹果将对ZFS的支持砍掉之后从工作了20多年的苹果离职，开了一家名为Ten's Complement的公司，该公司提供Z-410，较MacZFS提供更新更稳定的移植。

不过，砍了ZFS后的苹果目标也变得更清晰——和Sun的谈判让苹果觉得与其支付高额的协议费，还不如雇人自己做个新的，再说了，作为比Sun大得多的IT公司，苹果可以轻而易举地搞个更强大的东西灭了它，因为ZFS其实也不如传说中的那样好。

首先，时代在进步。ZFS之后，又有很多新的和文件系统相关的研究，如Ohad Rodeh的论文，即成为后来BtrFS实现的基础，可能比ZFS做得更好。

其次，ZFS是十年前开始设计的文件系统，但十年中，存储工具已发生了重大的变化。ZFS为传统磁盘设计，但传统磁盘的市场空间已不断被SSD、闪存的吞食。尤其是MacBook Air中使用的Flash存储器便宜好用又小巧，可能将来会在MacBook Pro甚至iMac中得到更大的推广。采用为传统磁盘优化的ZFS就不显得那么有吸引力。

最后，ZFS和苹果有不同的用户群。ZFS目标用户是大企业的工作站和服务器。在那里，大容量的存储空间、高级的卷管理显得非常重要，但苹果面对的基本都是个人用户——先前苹果还卖服务器，但后来Xserve都被苹果砍了。有几个个人用户需要使用到ZFS这些高级的功能呢？更重要的，苹果的主要利润将移到iPhone、iPod、iPad、Apple TV这些小设备上，ZFS需要占用大量的内存来实现文件系统操作，在这些小设备上，内存很少，ZFS根本跑不起来。

苹果非常清楚这些问题，工程师们现在一定在紧锣密鼓地开发下一代文件系统。在10.7及10.8中，这套文件系统并未浮出水面，但一些细节值得留意。在10.7中，苹果发布了Core Storage，但并未声张。这是一套逻辑卷管理工具，类似于前文提到的FreeBSD的GEOM。这个版本的File Vault 2亦使用Core Storage重写。可以看到虽然苹果在上层不断地淡化文件系统的概念，例如iCloud的发布和iOS中对于文件这一概念的故意忽略，但苹果在底层文件系统上的动作越来越大，想必在将来，苹果定会让我们感到重大的惊喜。

王越

美国宾夕法尼亚大学计算机系研究生，中国著名TeX开发者，非著名OpenFOAM开发者。

责任编辑：陈博（chenbo@csdn.net）

新书上架

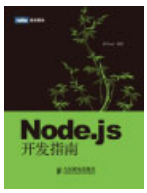
本月新书

Uninty 3D游戏开发

作者：宣雨松

出版社: 人民邮电出版社

本书介绍了Unity环境搭建、编辑器、GUI游戏界面、构建游戏脚本等知识，并以实例原型，向读者详细介绍游戏制作的整个过程。



Node.js开发指南

作者：郭家宝

出版社: 人民邮电出版社

不少开发者在入门Node.js时总要经历一个痛苦的思维转变过程，给学习带来巨大的障碍。而本书的目的就是帮助读者扫清这些障碍。

深入理解软件构造系统：原理与最佳实践

作者：Peter Smith

译者：仲田　等

出版社：机械工业出版社

本书对软件构造过程进行了彻底而深入的研究，其中包括在精心设计的构造过程中所做出的各种选择、相关利弊，以及遇到的难点。

神一样的产品经理——基于移动与互联网产品实践

编著：闫荣

本书由浅入深、循序渐进地阐述了产品经理、产品需求、用户体验、项目管理、产品运营和产品团队管理的内容，理论与实践相结合，尤以实践为重。

Oracle Database 11g RAC手册（第2版）

作者：K Gopalakrishnan，

译者：贾洪峰 梁涛 郭绍明

出版社：清华大学出版社

通过阅读本书，可以学习如何准备硬件、部署Oracle RAC、优化数据完整性和集成无故障转移保护。此外，还讨论了故障排除、性能调优和应用程序开发等内容。

推荐图书

Java7入门经典

作者： Ivor Horton

译者： 梁峰

出版社：清华大学出版社

一门语言的兴盛、发展，都离不开新鲜血液的加入，而新鲜血液的加入离不开基础教育的引导。《Java 7入门经典》这一类的书籍正是Java风靡世界的原动力。本书结构合理、表述精确、通俗易懂，消除了众多新入门者的陌生感和神秘感。

然而《Java 7入门经典》也有其局限性，这类书籍面向的是刚入门或者刚接触Java语言的人群，所以从基础的语言出发，基本在讲解语法和Java基础类库的层面。

本书只在第一章简单介绍了Java语言的概貌，就迅速地转向了对Java语言编程思想的介绍，避免了一般入门级别的书籍长篇大论地对Java的历史和成绩的吹嘘，规避了初学者在几个小时的阅读之后发现自己依然停留在语言介绍上的抓狂。

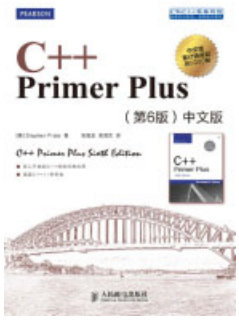
本书对Java语言的泛型单列一章进行讲解，泛型语法是Java 5才引入的，泛型的好处是在程序在编译时能检查类型安全，并且所有的强制转换都是自动和隐式的，能够提高代码的重用率，这是一项相当有用的功能。如今Java7已支持泛型创建时的类型推断，在泛型引入初期，创建对象例如：`List<String> result = new ArrayList<String>();`，在引入类型推断之后，变化如下：`List<String> result = new Array<>();` ，ArrayList可以推断出泛型类型，比较方便程序员使用。

通过对本书的学习基本上能了解Java的筋骨。但今天的Java 7已比Java诞生时成长了很多，Java 7在不断进化中引入了众多为热爱者所称赞的变化，本书碍于篇幅和定位，对引入的很多新特性无法提及和介绍，如闭包语言的引入、try-with-resource语法的改进、Switch支持String、支持空指针，捕获多种异常、二进制的字面值、创建泛型时的类型推断等新特性。

如果读者想更进一步地了解Java知识，例如虚拟机相关的实现、垃圾回收算法、虚拟机调优或JavaEE方面等Java的强项所在，建议读者选择其他进阶读物。

当然任何一本书都不可能面面俱到。作为一本入门级书籍，《Java 7入门经典》尽管没能更加深入地介绍Java及以外特性，但对Java核心的基本语法、使用最多的框架集合、线程池等的讲解深入浅出。尤其是对目前流行的XML的支持也专门进行了讲解，加上一些实战的例子，足以让人在短时间对Java有全面的认识。

在Java 7即将大行其道时，《Java 7入门经典》的推出，正可谓恰逢其时。时也，命也，让我们和即将通过本书成为Javaer的朋友们，为即将到来的Java 7时代和《Java 7入门经典》的诞生欢呼吧，拥抱Java。（**点评人：张方勇　Oracle甲骨文Java开发资深软件工程师**）



C++ Primer Plus

作者： Stephen Prata

译者： 张海龙 袁国忠

出版社： 人民邮电出版社

在计算机编程语言中，C++一直被认为是难度最大、最复杂、灵活性最高的一种，让很多初学者望而生畏。在众多的C++入门读物中，很多作者都希望能够抚平C++和读者之间的那道鸿沟。《Accelerated C++》、《C++ primer》、《C++ programming》无疑都是其中的优秀作品。《C++ Primer Plus》也是一本可以与上述书籍相媲美的优秀作品，本书作者曾参与编写了十多本图书，在C++教学方面具有丰富的经验。

这是一本循序渐进的C++读物。本书兼顾了C语言和C++的特性，从C语言和C++语言最基本的变量、语句、操作符、函数开始介绍，使得读者能够循序渐进、从零开始学习C/C++。这对于从未接触过C语言编程，而希望直接介入到C++语言的读者来说，是一个很好的学习方式。

这是一本深入浅出的C++读物。在介绍完基本的C/C++特性后，本书开始重点介绍C++的类的特性，使度过C/C++基础学习的读者，能够真正进入面向对象编程的领域。C++的面向对象特性——包括模板、类的继承、多态、异常处理等——一直是所有面向对象编程语言中最为复杂和灵活的，因此C++语言也被看作是编程语言的实验基地。但本书在介绍这些高级特性时，通过各种实例来进行讲解，使得本书深入浅出。

这是一本开发人员的参考书。本书虽然定位为C++语言的学习读物，但对很多C++特性做了详细的介绍，可以作为工程人员在开发时的参考书，书中对于C++编程设计的概念性指导，尤其是类的继承、多态、友元、异常处理等，对具体设计开发很有帮助。

浅尝辄止的阅读，并不能真正领会C++的真谛。因此，本书对读者也提出了两个要求。

知行合一。在学习编程的过程中，必须能够一边看书，一边动手，知行合一，才能有所收获。本书在每章都设计了很多习题，这些习题设计的都很务实、精巧，对这些习题进行编程练习，可以帮助领会C++编程的重点，避免理解的误区。

探微知注。本书主要面向初、中级的读者，但也详细介绍了C++的特性，对于模板、STL、异常、RTTI等都进行了详细讨论，且尽量地深入细节。编程是一种艺术，也是一种工程。作为工程，知其然，而不知其所以然，必然不能设计出真正优雅、高效而安全的程序。在学习过程中，本着“探微知注”的心态非常重要，编程之路漫漫而修远，务必上下而求索。

学习是一种不断地自我修炼和沉淀的过程，在这个过程中，无疑会遇到很多困难和挫折，也许需要反复阅读、反复练习，唯有不厌其烦、脚踏实地，才能真正在学习中得到收获。

（点评人：靳志伟 中国农业银行软件开发中心工程师）

全球排行榜

01 Cracking the Coding Interview: 150 Programming Questions and Solutions
02 HTML and CSS: Design and Build Websites
03 The Mobile Wave: How Mobile Intelligence Will Change Everything
04 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition
05 iOS Programming: The Big Nerd Ranch Guide, Third Edition
06 C Programming Language (2nd Edition)
07 Introduction to Algorithms
08 JavaScript: The Good Parts
09 Professional Android 4 Application Development
10 slide:ology: The Art and Science of Creating Great Presentations

天珑书局（中国台湾）

01 HTML5完美风暴
02 《超强图解》前进Android Market! Google Android SDK实战演练，2/e
03 精通正规表达式，3/e
04 iOS创意程序设计家——iPhone+iPad跨平台通用，3/e
05 JavaScript大全，6/e
06 JavaScript设计模式
07 iOS SDK开发范例大全
08 HTML5 CSS3精致范例辞典
09 Android系统原理深入解析
10 黑客列传——计算机革命侠客志，25周年纪念版

第二书店

01 Java编程思想（第4版）
02 数学之美
03 Java核心技术，卷1
04 ASP.NET MVC 3高级编程
05 PHP和MySQL Web开发（原书第4版）
06 Oracle DBA手记4：数据安全警示录
07 算法导论（原书第2版）
08 Node.js开发指南
09 Unity 3D游戏开发
10 鸟哥的Linux私房菜.基础学习篇（第三版）

GEEK产品

CYBORG M.M.O.7鼠标

赛钛客以R.A.T.7鼠标为基础原型，对部分细节进行改进升级，推出了全新的CYBORG M.M.O.7鼠标。以全新的外观设计、出色的自定义调节功能和数量众多的按键来征服游戏玩家。

U Transfer

这款国外概念U盘名为U Transfer，它自带一个触摸屏及USB接口，可以不需要电脑就实现U盘间的文件传输。使用时只需将两枚U Transfer首尾相连，对于手边没有电脑又急需传输文件的人来说，这是一个很实用的产品设计。

Philips FLUID手机

Philips为我们带来的力作——拥有流线、多变的外形设计的Philips FLUID。这款创新设计手机的确让人惊叹，可佩戴装饰，也可取下把玩。总之，FLUID是手机中饰品做得最好的，也是将科技应用最到位的。

Aire面具

Aire是一款获得红点大奖的设计，它的概念为在慢跑的同时能为你的iPhone充电。其设计师Lammoglia声称这个概念设计可以鼓励人们进行锻炼，你呼吸得越多，获得的能量也就越多。

Humminbird RF35 无线腕式声纳探鱼器

Humminbird RF35内置有GPS定位器，配备了1.25英寸显示屏，通过内置的远程声纳感应器能够识别出75英尺范围内所有鱼类的位置，相关数据能够实时传输至手表上。当你不钓鱼时Humminbird RF35还可以当作一款普通的手表使用，配备有逆光照明器以及高级脉冲感应器。

Moverio BT-100智能眼镜

科研公司Epson抢先在美国市场推出一款搭载Android系统的智能眼镜。这款眼镜采用经修改的Android 2.2系统，内建的微型投射技术能够在虚拟80寸屏幕播放影像，并支持3D显示和Dolby Mobile立体声技术。官方称它的续航力可达6小时，而内置储存容量则有1GB，并支持最高32GB microSD卡扩充。

名人堂

“开放源代码运动”领袖
Eric Raymond

文 / 陈粲然

黑客与作家

Eric Raymond于1957年12月4日出生在美国马萨诸塞州的波士顿，而波士顿正好就是黑客文化发源地MIT的所在，也是Richard Stallman发动自由软件运动的大本营。不过波士顿并非Raymond真正意义上的故乡，他从小随着父母在世界各地东奔西走，直到1971年才随家庭搬回宾夕法尼亚洲。

1976年，他开始接触“黑客”文化，他最初的编程经验来自原始的ARPANet。Stallman在20世纪80年代初启动自由软件运动时，Raymond很快受到感召，投奔Stallman的门下，成为FSF（自由软件基金会）最早的撰稿人之一。

Raymond在1982年完成了自己的第一个自由软件项目，他在1985年起辞去了工作，开始集中精力于编程、写书及一些技术评论。其核心著作——《黑客道简史》、《大教堂和集市》、《如何成为一名黑客》、《开拓智域》、《魔法大锅炉》被业界称为“五部曲”。

Raymond在1990年编写了Jargon File（被称为《新黑客辞典》，主要用于记录一些术语和行话，同时也记录了一些计算机史上的一些趣闻轶事，甚至是一些“八卦”。在这些散落的记录当中，我们不仅可以一窥计算机领域的发展，并能够发现大量有价值、有帮助的资源）。

Eric Raymond是一位卓越的黑客、作家和演说家

《大教堂和集市》的诞生

1993年，Raymond决定自己做一个邮箱系统，他学习了Linus的做法：Linus并不是从头开始写Linux的。Raymond首先选择了Popclient作为雏形，然后对它进行重写，并根据用户反馈快速地进行修改。在Raymond认为条件成熟后，他将Popclient更名为Fetchmail。

Fetchmail的成功让Raymond看到了Linux开发风格的巨大魅力，于是他写了《大教堂和集市》，想把自己的心得体会告诉更多人，这本著作在1997年的Linux大会上发表。

在《大教堂和集市》一书中，Raymond用对大教堂和开放集市两种模式的比喻，形象地将商业封闭软件和自由软件区分开来——“我一直想找一个比喻，能够强调我所发现的在两种开发模式中所存在的重要区别。一种是封闭的、垂直的、集中式的开发模式，反映一种由权利关系所预先控制的级权制度；而另一种则是并行的、点对点的、动态的开发模式”，Raymond表示，他希望《大教堂和集市》这本书有血有肉、富有刺激性并能够启发人们思索。

开放源代码运动

Raymond认为“自由软件（Free Software）”一词容易与对知识产权的敌意联系起来，几乎不可能让大企业的CEO和CIO们喜欢，更容易把投资者吓跑。于是，他在1998年2月3日，Raymond提出了“开放源代码”这一术语，并得到了与会人士的广泛认同。之后，Raymond和Bruce Perens创立了开放源代码促进会（Open Source Initiative），打起了开放源代码软件的大旗。

开放源代码软件被定义为其源码可以被公众使用的软件，并且此软件的使用、修改和分发也不受许可证的限制。开放源代码软件通常是有版权的，它的许可证可能包含这样一些限制：着意的保护它的开放源码状态，著者身份的公告，或者开发的控制。“开放源代码”正在被公众利益软件组织注册为认证标记，这也是创立正式的开放源码定义的一种手段。

Raymond在《开放源代码的FAQ》中指出：“开放源代码软件是自由软件的营销手段。它是对自由软件的支持，更倾向于注重实际效果而不是意识形态方面的大肆宣扬。成功的主体并没有改变，失败的态度和象征主义却变了。”

1998年5月7日，Corel公司宣布了其基于Linux的Netwinder网络计算机。1998年6月22日，IBM宣布它将出售并支持Apache，并作为它的WebSphere组件的一部分。1998年7月17日，Oracle和Informix宣布它们将把数据库移植到Linux上。此后，许多IT业巨头，包括CA、IBM、Interbase、Sybase、HP和Sun，纷纷宣布了它们支持开源软件的计划。

Ra mond的失误

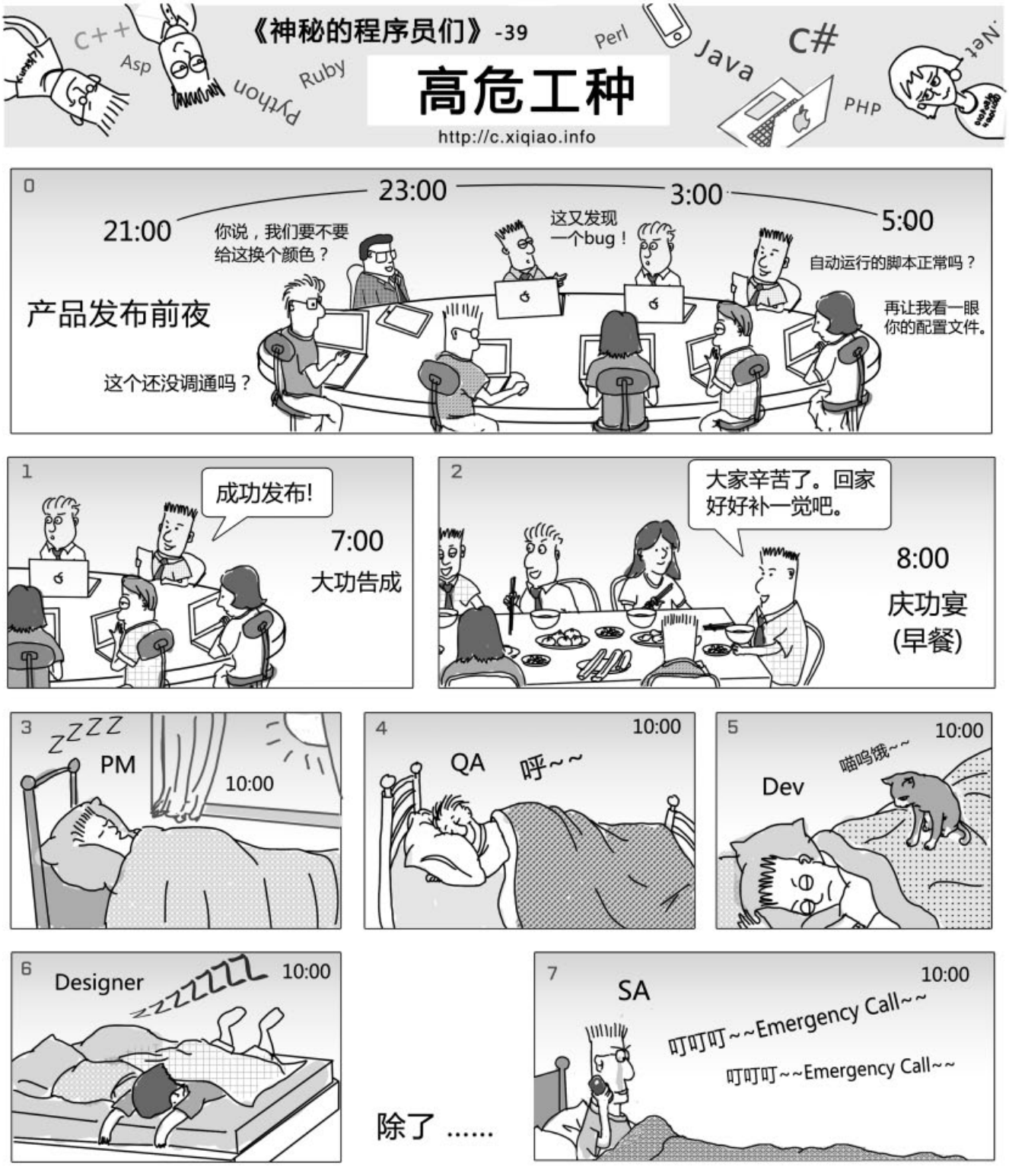
Raymond本人精通C、Lisp、Pascal、APL、Fortran及BASIC语言，曾在Apple II、Macintosh、Sun、IBM PC、VAX II-75、DEC-10、PDP-8、Z80等多种系统上开发过软件。他是Intercal编程语言的主创之一。曾经为Emacs编辑器的发展作过贡献，管理着30多个开源软件，以及10多个主要的FAQ。

比较有意思的是，Raymond曾花了两年时间编写了一个用于配置Linux内核的智能配置程序——CLM2，该程序可以智能地将troff标记页转换成XML DocBook。Raymond曾在采访表态说CLM2是他最好的作品之一，他用少于8000行的Python代码就完成了这个小型的系统。它会根据规则执行智能推理，确保得到的内核配置都是有效的。CLM2得到了Linux内核配置组的完全认可和Linus的批准，却被内核列表策略抢走了风头。

Raymond把这次失败归结于战略上的失败：“因为Linus放弃了他的领导职位，违背了他的承诺。开放源码文化在某些方面非常保守，会把好的作品关在门外。这是一个长期问题，我不知道将要如何解决。我还认为发生这种情况的部分原因是内核列表上的作者不喜欢让所有人都可以访问内核配置的想法，他们希望它继续成为一种魔法。”

整件事与Raymond提出的“在开放源码社区中，最好的代码会获胜”的理念相左，这件事令他非常沮丧。

如今，Raymond作为开放源代码促进会对开源媒体、商界及主流文化的形象大使与思想家，仍在世界各地坚持不懈地推动开发源代码软件运动。



西乔
设计师，项目经理。
2006年起携创业团队从事Web技术外包开发及产品咨询顾问。

如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享。
请给西乔发邮件: arthur369@gmail.com